Sistema de alertas PcComponentes
PcComponentes alert system

Por
Sergio Díaz Renedo
Miguel Neira Martín
Darío Mijares Muñiz

UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Ingeniería Informática, Ingeniería del Software
Facultad de Informática

Mercedes García Merayo

Madrid, 2019–2020

Agradecimientos

Quiero empezar agradeciendo, a mis amigos y compañeros de la carrera. En especial, a Iván y Guillermo, por toda su ayuda estos años. Sin ellos no habría llegado hasta aquí.

Por otra parte quiero agradecer a mi madre, a mi padre y a mi hermana. Por ser las personas que mas me han apoyado. Sobretodo en los momentos malos.

Darío Mijares Muñiz

Agradezco sobremanera a mis familiares cercanos: Miguel, Paloma y Beatriz ya que sin su apoyo y ánimos no hubiera sido capaz de superar la presión ejercida por el exceso de asignaturas estos dos últimos años ni hubiera podido mantener el ritmo para superar el grado.

A mis amigos que, aunque no comparten grado conmigo, han sabido apoyarme y ayudarme en lo que necesitaba, además de mejorar mi ánimo cada vez que nos reunimos.

Y, finalmente, gracias a mis compañeros de proyecto Sergio y Darío por tener paciencia en el desarrollo del mismo, así como aportar opiniones y valoraciones a lo largo de estos meses.

Miguel Neira Martín

Agradecer a mi familia y amigos el apoyo recibido durante los años de carrera, además de a los compañeros con los que he compartido asignaturas y proyectos, e incluso trabajo, y que hicieron más ameno este importante viaje.

También agradecerles a Darío y Miguel la implicación que han tenido en el proyecto, lo que nos ha permitido marcar objetivos mensuales que hemos cumplido hasta la fecha de entrega del mismo.

Sergio Díaz Renedo

A la Universidad Complutense de Madrid y a la Facultad de Informática por las instalaciones, su plan de estudios y los profesores que nos han impartido las asignaturas. Sin ellos no hubiera sido posible llegar hasta aquí.

Y, en especial, a Mercedes por darnos la oportunidad de trabajar en este proyecto y

por su implicación. Ha sido como trabajar con uno más. Muchas gracias.

Los autores

Sobre TEFLON

Teflon(cc0 1.0(documentación) MIT(código))es una plantilla de LaTeX creada por David Pacios Izquierdo con fecha de Enero de 2018. Con atribuciones de uso CC0.

Esta plantilla fue desarrollada para facilitar la creación de documentación profesional para Trabajos de Fin de Grado o Trabajos de Fin de Máster. La versión usada es la 1.3.

V:1.3 Overleaf V2 with pdfLaTeX, margin 1in, NO-bib

Contacto
Autor: David Pacios Izquiero
Correo: dpacios@ucm.es
ASCII: asciifdi@gmail.com
Despacho 110 - Facultad de Informática

Índice general

Página

Resumen

Este documento tiene como objetivo informar del desarrollo del proyecto que ha dado lugar al sistema de alertas de PcComponentes [23].

Nuestro propósito es facilitar la compra de artículos a los usuarios, alertándoles si se ha producido una oferta en aquellos productos por los que han mostrado interés.

PcComponentes es una tienda de tecnología situada en Murcia y en Madrid que posee una página web para realizar compras en línea. Dada la cantidad de productos a la venta, las rebajas que realizan habitualmente y la importancia del portal a nivel nacional, sería interesante poder llevar a cabo un seguimiento de los componentes que nos interesan, recibiendo avisos cuando se reduzcan los precios a umbrales que especifiquemos, siendo este un servicio que no ofrece la propia compañía.

En la competencia es posible encontrar algún servicio que - ya sea desarrollado por la propia empresa o por otros - permita realizar un seguimiento de los precios de aquellos artículos por los que se muestra interés. Amazon es un gran ejemplo: existen diversas aplicaciones que facilitan historiales de precios, ofertas, notificaciones, etc. como Camel [28] o Price Tracker for Amazon [25].

El trabajo de fin de grado consiste en el desarrollo de una aplicación en Android [29] a través del entorno de programación Android Studio [7]. Para ello, en primer lugar, se realiza una recogida de los componentes que aparecen en la web de PcComponentes mediante *web scraping*. Esta información se almacena en una base de datos de Firebase [12], que nos permitirá notificar a los usuarios los componentes en los que estaban interesados que han bajado su precio del umbral indicado. La comunicación se realizará a través de correo o *push* del teléfono según las preferencias de los usuarios.

Abstract

In the present document the objective is to inform and clarify the idea behind the developed project and how the PcComponentes alert system has been implemented.

Our purpose is to facilitate the purchase of components, alerting the users if there is a sale in those products they have shown interest.

PcComponentes is a technology store located in Murcia and Madrid that has a website to make purchases online. Given the number of products for sale, the discounts they usually make and the importance of the portal, it would be interesting to track components and receive notifications when prices are below a certaint quantity, specified by the user, which is a service not offered by the company itself.

It is possible to find a service that - whether developed by the company itself or by others - allows you to track products and prices for those items in which you are interested. Amazon is a great example: there are several applications that provide price histories, sales, notifications, etc. such as Camel and Price Tracker for Amazon.

This project consists in the design and implementation of an application in Android using Android Studio. To do this, the first thing is getting the collection of components stored in PcComponentes website using web scraping. This information will be stored into Firebase database, which will allow us to notify users, via email or *push*, acording to their preferences, about components they are following and are in sale.

Capítulo 1

Introducción

PcComponentes es una tienda online que se encarga de vender artículos de tecnología, especialmente los referidos a componentes y periféricos de ordenador. Su comercio siempre fue online hasta hace pocos años, cuando estableció dos tiendas físicas en Murcia y Madrid. Es la empresa líder en España en este sector gracias a un buen servicio técnico y a las facilidades que proporciona para que el usuario pueda configurar su propio ordenador mediante el configurador de PCs que tiene en su página web.

Debido a la alta frecuencia de ofertas que lanzan, los usuarios parecen sentirse obligados a visitarla diariamente para comprobar si los productos que desean se han rebajado de precio. La idea del proyecto nace en base a esto: conseguir que los usuarios de la página puedan saber si los productos que quieren comprar se han rebajado de precio sin necesidad de conectarse de forma diaria e ir buscando dichos productos uno a uno.

Muchos portales existentes utilizan esta funcionalidad. Por ejemplo, Steam [4] (tienda muy importante de videojuegos en ordenador) manda notificaciones vía email a sus usuarios cada vez que alguno de los productos que tienen en la lista de deseados son rebajados. Resulta muy útil para los usuarios y ayuda a aumentar las ventas al proporcionar de forma automática dicha información a sus clientes.

Para ofrecer este servicio, el grupo del proyecto acordó realizar una aplicación móvil en la que los usuarios pudieran escoger los artículos en los que estaban interesados, estableciendo un precio a partir del cual ser notificados. De esta forma, es posible enviar notificaciones a aquellas personas interesadas en componentes cuyo precio ha disminuido por debajo del umbral establecido o informarles de que han sido eliminados de la tienda.

La elección de una aplicación móvil viene dada por el gran uso de dispositivos móviles en la sociedad, lo cual hace que sea mucho más accesible por la mayoría de usuarios, permitiendo utilizar el programa en cualquier momento y recibir las notificaciones de manera instantánea. Se trata de una solución cómoda, rápida y eficaz.

Respecto a las notificaciones, se decidió que el usuario pudiese decidir recibirlas de dos

formas. La primera, a través de una notificación móvil de tipo *push*, similar a las que utilizan servicios como WhatsApp o Gmail. La segunda, mediante correo electrónico. En ambos casos se realizan avisos distintos cuando se eliminan artículos de interés o se producen rebajas en algunos de ellos.

Para conseguir los datos de la página web de PcComponentes se utilizó *web scrapping*. *Web scrapping* es una técnica que utilizan programas de software para recoger los datos de una web. En nuestro caso, captura todos los productos de la página y de cada uno extrae el código de producto, su nombre, precio, enlace a la imagen, categoría y enlace a la pagina del artículo. No consideramos necesario utilizar más datos ya que hemos destinado el proyecto a las notificaciones de ofertas y no a explorar los componentes del portal.

En lo relativo a la base de datos, tras utilizar inicialmente MySQL [18] sobre una instancia RDS de Amazon AWS [2], se decidió usar, finalmente, Firebase. Se trata de una plataforma en la nube desarrollada por Google, que proporciona múltiples facilidades para el desarrollo de aplicaciones móviles. Entre las funcionalidades que ofrece, las más importantes para este proyecto son la base de datos no relacional, un sistema de autenticación y un servicio de notificaciones *push* y de envío de correos a los usuarios, para diversas funcionalidades como confirmar la dirección en el registro, modificar la contraseña, etc.

En cuanto al sistema de inicio de sesión y registro de usuario, queríamos tener la mínima información posible por protección de datos. Inicialmente, en MySQL, almacenábamos su correo electrónico junto con su contraseña tras aplicar alguna función resumen (*hash*) siguiendo los estándares de seguridad recomendados. Finalmente, con Firebase, la gestión de las contraseñas y credenciales se realizan de manera transparente.

Para distribuir el desarrollo del proyecto consideramos tres bloques. El primero, la recogida de datos de la web (*web scrapping*) y el envío de notificaciones, fue asignado a Miguel Neira Martín. La segunda, corresponde a la aplicación móvil de los usuarios y el despliegue de Firebase, de la que se ha responsabilizado Sergio Díaz Renedo. Por último, la visualización de los componentes y el registro del interés de los usuarios, ha sido realizada Darío Mijares Muñiz.

En el desarrollo de la aplicación no hemos buscado la complejidad de funcionalidades y vistas. Creemos que es importante facilitar a los usuarios la información de las ofertas en componentes de interés a través de un acceso rápido y que, con el menor número de pasos, sean capaces de añadir a sus listas productos en los que están interesados así como los umbrales de los precios de los mismos.

Gracias a optativas como “Programación de Aplicaciones para Dispositivos Móviles (PAD)” y “Gestión de Información en la Web (GIW)” hemos tenido facilidades para realizar la aplicación móvil y el proceso de recogida de información de componentes y de notificaciones. Aun así, hemos necesitado conocimientos adicionales proporcionados por documentaciones oficiales u obtenidos de foros como *Stack Overflow* [33].

1.1. Objetivos

El objetivo global del trabajo es el desarrollo de una aplicación móvil que permita la gestión de alertas de reducción de precios de componentes de la web PcComponentes, seleccionados por los usuarios. Para ello la aplicación debe ofrecer las siguientes funcionalidades.

Usuarios

- Registrarse en la aplicación.
- Iniciar sesión en la aplicación.
- Navegar por el listado de componentes
- Añadir a la lista de interés componentes y fijar un precio umbral de notificación.

Proceso remoto

- Recoger y actualizar la información de los componentes periódicamente.
- Enviar notificaciones, según las preferencias de los usuarios, cuando los componentes en los que están interesados alcancen un precio por debajo del indicado.
- Mantener actualizada la bitácora.
- Eliminar toda información relacionada con componentes descatalogados y eliminados de la página web.

Capítulo 2

Análisis de requisitos

A continuación se van a detallar diferentes puntos que necesitamos que se cumplan para el correcto funcionamiento del proyecto.

2.1. Requisitos

2.1.1. Registro

- Se requiere conexión a Internet
- El usuario debe introducir un correo electrónico y contraseña válidos
- El correo electrónico no puede haber sido registrado anteriormente

2.1.2. Login

- Se requiere conexión a Internet
- Los datos introducidos por el usuario deben ser correctos
- El correo electrónico debe haber sido verificado desde el enlace recibido por email

2.1.3. Aplicación

- Se requiere conexión a Internet
- La lista de productos debe mostrar al menos su nombre, precio e imagen.
- Debe haber una lista para los productos seguidos por el usuario.

- El precio de seguimiento de un producto debe ser inferior al precio del producto y mayor que cero.
- El usuario puede dejar de seguir un producto.
- El precio de seguimiento de un producto puede ser actualizado por un usuario.
- Existirá un buscador para facilitar al usuario la busque los productos que desee.
- Habrá una opción de perfil donde el usuario podrá, entre otras cosas, elegir qué notificaciones quiere recibir.
- Habrá una pestaña para que el usuario pueda cerrar la sesión.

2.1.4. Proceso remoto

- Para enviar notificaciones, tiene que haber usuarios interesados en componentes y que sus precios bajen del umbral proporcionado o sean eliminados de la base de datos.
- Dado que el proceso se basa en los elementos HTML de las páginas de PcComponentes para recoger toda la información de los componentes, es necesario que la propia tienda no cambie el código de su aplicación web (nombres de clases CSS, distribución de etiquetas HTML, etc.).
- Para enviar notificaciones por email es necesario que los usuarios activen la opción de recibir correos en la aplicación.
- Para eliminar un componente y toda la información relacionada (listas de interés), se debe cumplir que el programa de recogida no lo encuentre en la página de PcComponentes.

Capítulo 3

Entornos de Desarrollo

En este capítulo se van a enumerar una serie de librerías, herramientas, entornos y lenguajes de programación que hemos utilizado y han facilitado la realización del proyecto. También se procederá a explicar nuestro sistema de control de versiones y qué base de datos usamos, además de utilidades para prototipado de las interfaces.

3.1. Frameworks y Librerías

Empezando por la lista de entornos y librerías utilizados, encontramos los siguientes:

3.1.1. Jupyter Notebook

Jupyter Notebook [27] surgió del proyecto *Jupyter* con el objetivo de "desarrollar software de código abierto, estándares abiertos, y servicios para la informática interactiva a través de docenas de lenguajes de programación". Es un entorno que facilita la programación con Python [37] y, además, se encuentra incluido dentro de la distribución Anaconda Python 3 [1]. Su uso ha facilitado la realización de la recogida de componentes en la base de datos y el envío de notificaciones, siendo posible exportar fácilmente el código para poder ser ejecutado en la máquina Linux EC2 de Amazon AWS cada cierto tiempo.

3.1.2. XAMPP

XAMPP [39] es un paquete con aplicaciones como Apache, MySQL, FileZilla, Mercury y Tomcat para Windows. Su uso nos ha proporcionado facilidad en la realización de pruebas locales, gracias a Apache y MySQL, sobre la base de datos antes de ejecutar remotamente el programa de recogida de información.

3.1.3. Microsoft Visio

Microsoft Visio [32], propiedad de Windows, es un entorno destinado a la realización de diferentes diagramas y, así, poder realizar fácilmente el diagrama entidad-relación de la base de datos de una manera más visible.

3.1.4. Android Studio

Android Studio, desarrollado por Google, es el principal entorno de desarrollo para aplicaciones móviles, con soporte para diversos lenguajes de programación como Java [15], C++ y Kotlin [17].

3.1.5. PuTTY

PuTTY [8], desarrollado por Simon Tatham, es un cliente que permite realizar conexiones SSH para así establecer un enlace entre nuestras máquinas y la EC2 de Amazon AWS remotamente y facilitar el uso del terminal de la misma e introducir comandos para, por ejemplo, configurar Crontab [5].

3.1.6. WinSCP

WinSCP [38], desarrollado por Martin Prikryl, es un cliente que facilita realizar una conexión SFTP, en nuestro caso, con la máquina EC2 de Amazon AWS y poder manipular el sistema de ficheros de la misma fácilmente en modo gráfico.

3.1.7. Beautiful Soup 4

Beautiful Soup 4 [3] es una librería de Python que facilita la lectura de datos de textos HTML y XML, que nos permite procesar el HTML de las páginas pertinentes de PcComponentes y navegar a través de la información que interese.

3.1.8. MySQL Connector

MySQL Connector [19] es una librería de Python 3 que permite realizar la conexión a base de datos MySQL desde nuestro código, ya sea local o remotamente, proporcionando herramientas para ejecutar sentencias SQL. Fue utilizado antes de migrar a Firebase.

3.1.9. Urllib

Urllib [36] es una librería de Python 3 que permite manejar enlaces a través de utilidades como Request para conectarse y obtener el HTML de una página y Parse para codificar los argumentos de las peticiones de los enlaces.

3.1.10. Crontab

Crontab es un programa de Linux que permite la ejecución de tareas en ciertos periodos de tiempo dados, ya sea en una hora o fecha concreta o, por ejemplo, cada 15 minutos. Ésto hace que podamos realizar la ejecución del script de recogida de componentes fácilmente, así como volcar en una bitácora los momentos de ejecución y lo que tardan en ejecutarse las tareas.

3.1.11. MySQL Workbench

MySQL Workbench [20] es un entorno que permite el diseño y gestión de bases de datos. Fue muy útil para conectarse a la máquina RDS de Amazon AWS que contenía almacenada la información de los componentes y usuarios antes de migrar a Firebase.

3.1.12. SMTPlib

SMTPlib [30] es una librería de Python 3 que proporciona utilidades para conectarse a algún servidor SMTP y mandar correos. En nuestro caso utilizamos el servidor SMTP proporcionado por Gmail [11] para poder notificar a los usuarios de la bajada del precio o eliminación de componentes si eran de su interés.

3.1.13. Email.message

Email.message [10] es una utilidad encontrada dentro de la librería email [9] de Python 3 que permite gestionar fácilmente el contenido de los correos definiendo destinatarios, emisores, cabeceras y texto. A través del servidor SMTP de Gmail se mandan los objetos generados por esta librería.

3.1.14. Time

Time [35] es una librería de Python 3 que ayuda a utilizar funciones relacionadas con el tiempo. En nuestro caso es útil para medir cuánto tiempo tarda en ejecutarse el proceso

remoto de recogida de componentes y envío de notificaciones.

3.1.15. Datetime

Datetime [6] es una librería de Python 3 que proporciona herramientas para la manipulación de fechas. En nuestro caso se utiliza para, junto a la librería time, añadir a los archivos de bitácora información del tiempo que tarda en ejecutarse el proceso cada vez que se inicia automáticamente gracias a Crontab.

3.1.16. Firebase_admin

Firebase_admin [13] es una librería de Python 3 que proporciona diferentes módulos para autenticar, conectar y operar sobre datos encontrados en la base de datos facilitada por las credenciales introducidas y así obtener una referencia al cliente que permita las operaciones anteriores.

3.2. Lenguajes de Programación

Durante la realización del proyecto se han utilizado los lenguajes de programación presentados a continuación.

3.2.1. Python

Python es un lenguaje de programación multiparadigma cuyo objetivo es la claridad del código. Es uno de los más utilizados actualmente debido a su baja curva de aprendizaje, existencia de numerosas librerías para fines muy dispares y facilidad en el entorno *Big Data*.

3.2.2. Kotlin

Kotlin es un lenguaje de programación desarrollado por JetBrains en 2010 y liberado bajo la licencia Apache 2 en 2012. En 2017, Google lo nombró lenguaje oficial para desarrollos Android [16].

3.2.3. Java

Java es un lenguaje de programación comercializado en 1995 por Sun Microsystems. Es considerado el lenguaje principal de la programación orientada a objetos, aunque haya sido desplazado por Kotlin en el desarrollo de aplicaciones móviles.

3.2.4. MySQL

MySQL, funcionando sobre SQL, es uno de los sistemas de gestión de bases de datos relacionales más populares junto con los de Oracle y Microsoft. Fue utilizado en el proyecto antes de realizar la migración a Firebase.

3.3. Base de Datos

A pesar de inicialmente haber optado por utilizar MySQL sobre una máquina RDS de Amazon AWS, a medida que el proyecto fue desarrollándose, decidimos realizar una migración a Firebase debido a su comodidad para integrarse en apps móviles realizadas con Android Studio.

Inicialmente, la aplicación requería una base de datos MySQL almacenada en los servicios de Amazon RDS de Amazon AWS ayudándonos de MySQL Workbench si era necesario. Además, dado que los móviles no son tan potentes como los equipos de escritorio, era arriesgado realizar una conexión directa a la base de datos con librerías que así lo permiten, ya que podía llegar a saturar la aplicación. Para ello, se optó por desarrollar un pequeño servidor web que funcionaba a modo de API, escuchando peticiones y realizando operaciones como escribir datos, devolver información en formato JSON, etc. En la Figura 3.1 se puede contemplar el diseño entidad-relación inicial de la base de datos.

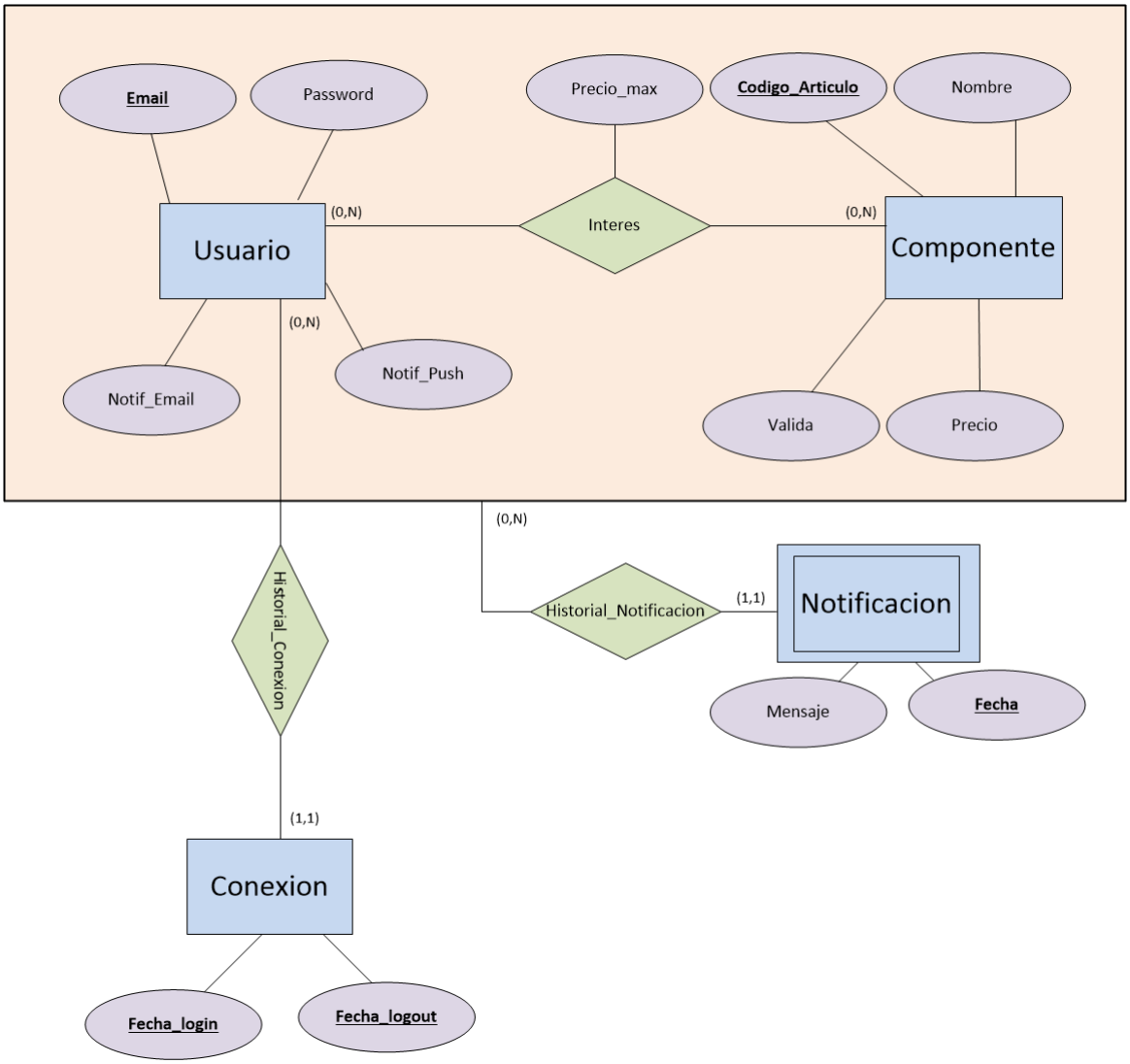

Figura 3.1: Base de datos MySQL inicial

Tras la migración a Firebase se tuvo que adaptar el proceso remoto Python de recogida de componentes para que se almacenase la información en la base de datos de Firebase a través del módulo firebase_admin que permite, como se ha visto anteriormente, autenticarse y operar sobre los datos.

Las colecciones presentan documentos para usuarios y componentes con una estructura como la que se presenta en las Figuras 3.2 y 3.3.

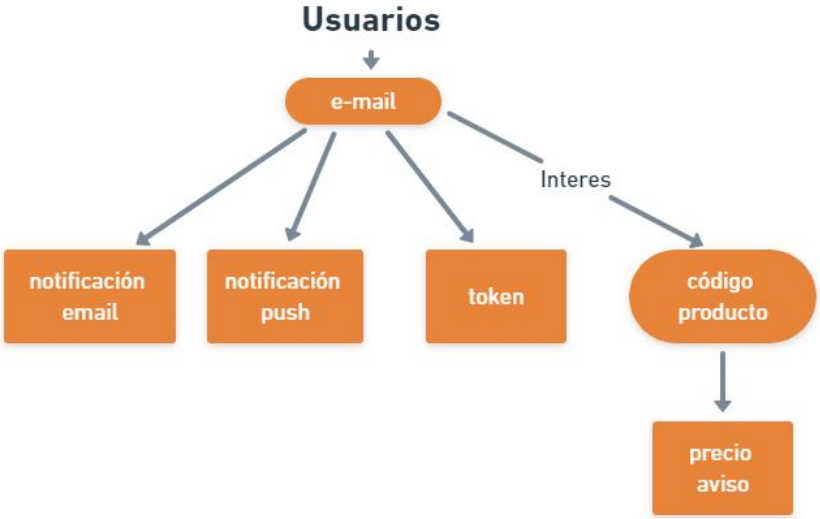

Figura 3.2: Base de datos colección usuarios

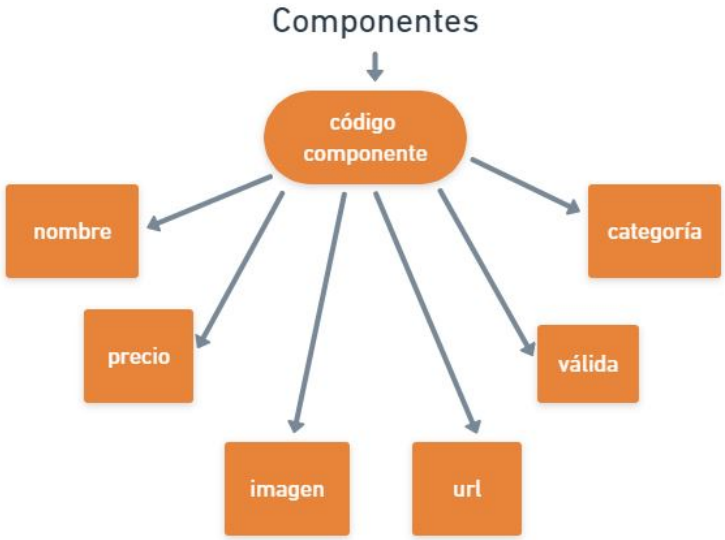

Figura 3.3: Base de datos colección componentes

3.4. Control de Versiones

Con el objetivo de desarrollar paralelamente el proyecto y para mantener un correcto control de versiones, se decidió hacer uso de la plataforma GitHub [24].

GitHub ofrece un servicio, tanto gratuito como de pago, para el control de versiones de proyectos usando la tecnología Git [31], registrando cada cambio subido al servidor gracias a su almacenamiento en la nube, permitiendo volver a versiones anteriores o comparar los cambios realizados entre las mismas.

Para ello creamos dos repositorios, uno para el desarrollo de la aplicación móvil y otro para

el web scraping y notificaciones, con una forma tal y como se aprecia en las Figuras 3.4 y 3.5 a continuación:

- Aplicación: https://github.com/Sergidia/PcComponentesTFG

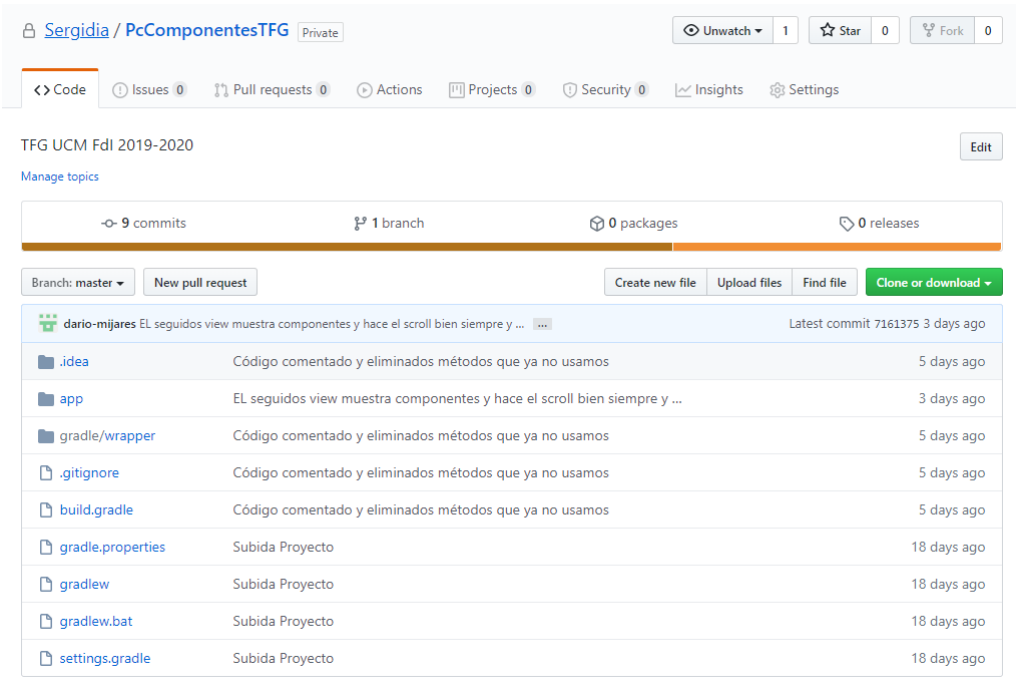

Figura 3.4: Repositorio del código de la aplicación

- Web scraping: https://github.com/Sergidia/PcComponentesTFG_WebScrap

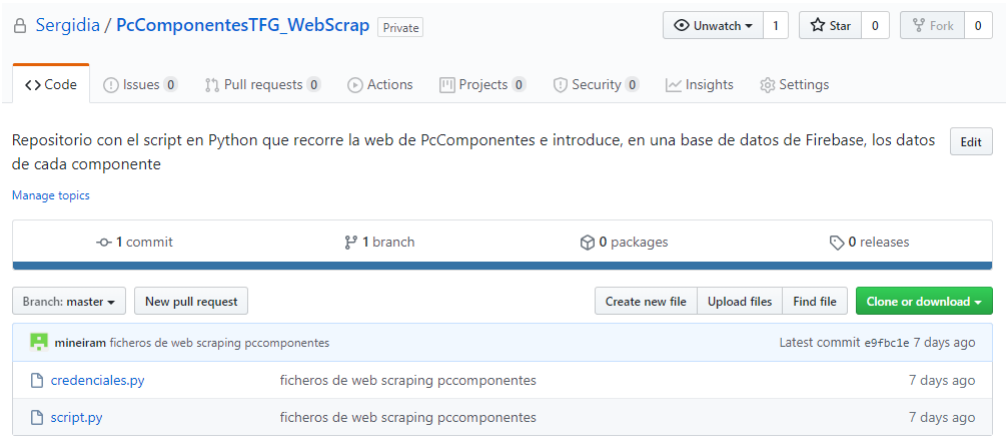

Figura 3.5: Repositorio del código del web scraping y notificaciones

3.5. Maquetación y Prototipos

Para la parte de Maquetación y prototipos se crearon unos MockUps para que ayudaran a los diseñadores de la aplicación a tener una idea general del aspecto que debía tener, con el objetivo de resultar amistosa para el usuario. Estos fueron creados a través de la herramienta web moqups [22]. A continuación se muestran los prototipos de la interfaz (Figuras 3.6, 3.7 y 3.8).

Figura 3.6: Boceto pantalla de registro

Figura 3.7: Boceto pantalla de login

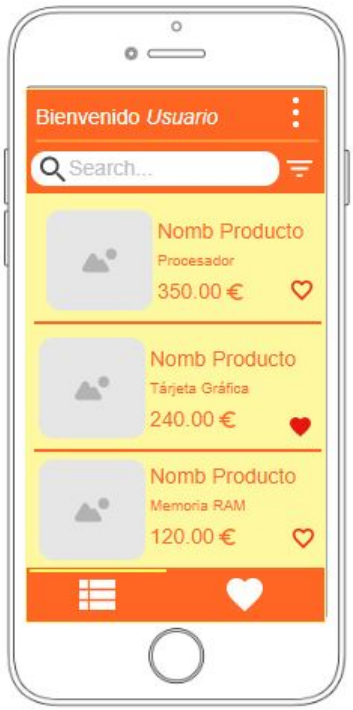

Figura 3.8: Boceto pantalla de inicio

Capítulo 4

Diseño e Implementación

En este capítulo se presentarán diferentes fases del proyecto para poder comprender por qué utilizamos las tecnologías mencionadas y reproducir la ejecución del mismo a través de un manual de instalación y configuración.

4.1. Fase de análisis

Una vez planteado el proyecto se analizaron diferentes tecnologías y lenguajes de programación para afrontar su implementación, desde extensiones para Google Chrome [21] con tecnologías web hasta una app de móvil, usando lenguajes como Java, PHP, Python y Kotlin y bases de datos tanto relacionales como no relacionales.

Finalmente, como se ha descrito en las páginas anteriores, se decidió hacer una aplicación móvil y no una extensión de Chrome por los problemas relacionados con su publicación en Chrome Web Store. En cuanto a la base de datos, la decisión inicial de hacer uso de una base de datos relacional en MySQL fue reemplazada por el uso de una no relacional que ofrecía Firebase y que proporcionaba más seguridad para el almacenamiento de contraseñas y las claves de *hashing*. En lo referente a los lenguajes de programación, se decidió usar Python para el script de recogida de componentes mediante web scraping, PHP para el envío de las notificaciones y Java y Kotlin para el desarrollo de la aplicación.

4.2. Fase de investigación

Una vez elegidas las tecnologías a usar en la fase de análisis nos encontramos ante algunas, como los lenguajes Kotlin o Python, el servicio de Firebase y la aplicación Android Studio, que desconocíamos al no haberlas estudiado durante el grado. Por ello, debimos formarnos personalmente y aprovechamos la ocasión para matricularnos en la asignatura optativa de "Programación de Aplicaciones para Dispositivos Móviles" y "Gestión de Información

en la Web", lo que nos ayudaría en el proceso.

4.3. Fase de Instalación y Configuración

En esta fase se van a describir los pasos a seguir para recrear y ejecutar el proyecto.

Paso 1 - Instalación de Android Studio: Se descargó la aplicación de código abierto oficial de Android para el desarrollo de la aplicación móvil.

Paso 2 - Creación y puesta a punto de la máquina EC2 de Amazon AWS: Para permitir la ejecución del proceso automático de recogida de componentes y notificación de rebajas en artículos de interés, es interesante utilizar una máquina virtual de la nube de Amazon que esté activa siempre que se pueda y funcione sobre Ubuntu.

Es necesario iniciar sesión o registrarse en Amazon AWS [1]. Si es la primera vez que se accede, es posible obtener 12 meses de acceso a la capa gratuita del servicio, suficiente para mantener una de estas máquinas activa las 24 horas.

Una vez esté la sesión iniciada, habrá que entrar en la consola. A continuación, se debe acceder a la pestaña de "Servicios" y entrar en "EC2" (ver Figura 4.1), lo cual permitirá tener una vista general de gestión de las instancias de este tipo.

Figura 4.1: Acceso a EC2

Seleccionando "Launch instance" será posible crear una máquina siguiendo una serie de pasos:

- Para el primer paso, se ha seleccionado la imagen "Ubuntu Server 18.04 LTS (HVM), SSD Volume Type".
- En el segundo paso, para aprovechar la prueba gratuita suficiente para hacer pequeñas pruebas, se deberá escoger la instancia del tipo "t2.micro". Tras esto, puede seleccionarse "Review and Launch".

[1] https://aws.amazon.com/es/

- Si inicialmente no hay ningún grupo de seguridad creado, volvemos al paso 6 "Configure Security Group" y nos aseguramos de que el campo "Source" marca "My IP" o "Custom" y "0.0.0.0/0", como se muestra en la Figura 4.2 (de este último modo, podría acceder cualquiera). "Type" debe tomar el valor "SSH" para permitir el acceso a la instancia posteriormente.

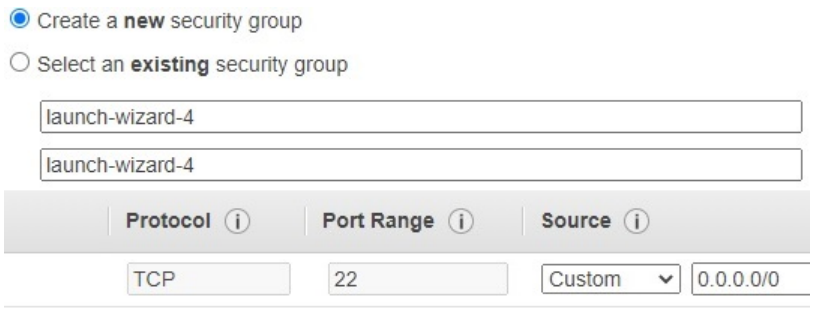

Figura 4.2: Configuración del grupo de seguridad

- Finalmente, se procederá a hacer una revisión de la configuración, para terminar la creación haciendo clic sobre "Launch".
- Una vez creada la instancia, hay que esperar hasta que ésta sea puesta en marcha.
- Cuando el estado de la instancia sea "running" quiere decir que ya es posible conectarse desde cualquier protocolo permitido (SSH, SFTP) tal y como se ve en la Figura 4.3.

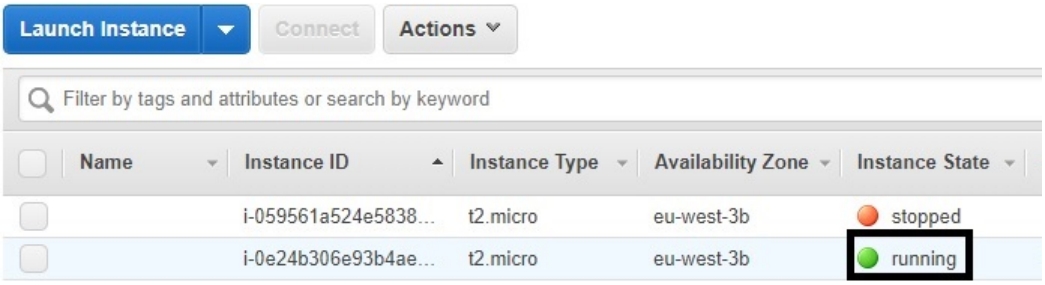

Figura 4.3: Instancia EC2 "running"

Para realizar la conexión a la instancia y trabajar cómodamente sobre la máquina, utilizaremos PuTTY ya que permite actuar fácilmente desde Windows. Para ello, será necesario proporcionar una serie de datos para completar el protocolo SSH.

Una vez se encuentre PuTTY instalado en el equipo habrá que seguir los siguientes pasos para configurar correctamente la conexión [2]:

- Descargar el par de claves. Si no se tienen, acceder al apartado "Key Pairs" en el apartado de Amazon EC2 y crear un nuevo par, seleccionando el formato ppk para que sea compatible con PuTTY.

[2] *Conexión a la instancia de Linux desde Windows mediante PuTTY*

- Ejecutar PuTTY:
  - En “Session”, rellenar el campo “Host Name” con la dirección obtenida en el apartado “Instance” de Amazon AWS EC2 en Public DNS (IPv4). En la Figura 4.4 se muestra un ejemplo para la dirección “ec2-52-47-193-84.eu-west-3.compute.amazonaws.com”. Además se debe indicar en el campo “Port” el valor 22 y seleccionar como “Connection type” el tipo SSH.

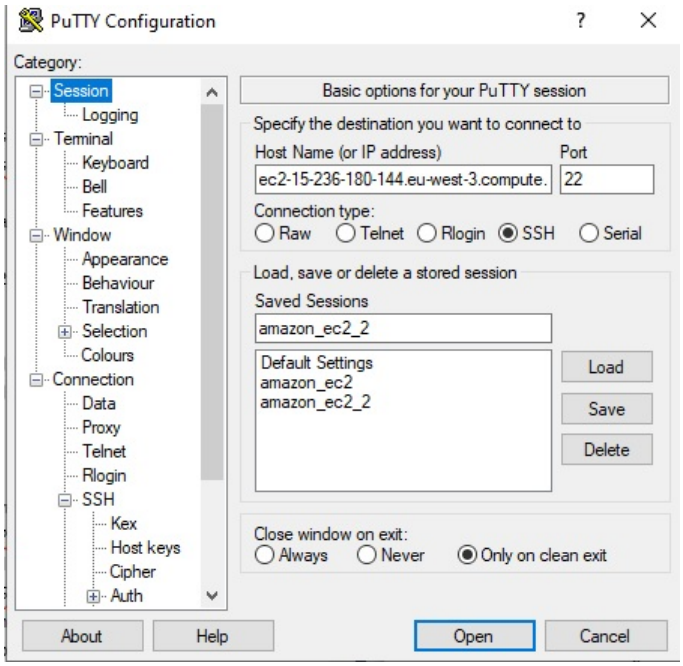

Figura 4.4: Configuración de sesión de PuTTY

  - En “Connection”, expandir “SSH” y elegir “Auth”. Seleccionar “Browse” para así importar el archivo con extensión ppk obtenido previamente al crear el par de claves (ver Figura 4.5).

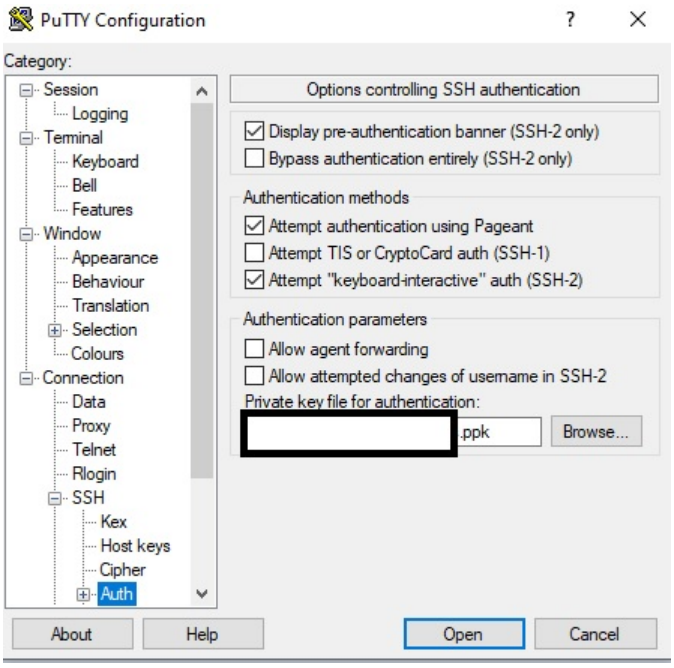

Figura 4.5: Configuración de autenticación de PuTTY

- Para guardar los datos de conexión, seleccionar "Session" de nuevo y, en "Saved Sessions", introducir un nombre en el primer campo de texto y pulsar el botón "Save".
- Seleccionando "Open" será posible realizar la conexión.
- Introducir como usuario en la consola "ubuntu". Ya será posible ejecutar comandos sobre la máquina.

Para permitir una cómoda transferencia de archivos desde Windows, se ha optado por utilizar WinSCP.

Para su correcto funcionamiento, es necesario tener en cuenta la siguiente secuencia de pasos [3]:

- Ejecutar WinSCP.
- Asegurarse de que "Nuevo sitio" está marcado en la parte izquierda de la ventana "Iniciar Sesión". Rellenar el apartado de "Sesión" con protocolo SFTP, nombre o IP del servidor de nuestra máquina, puerto 22 y usuario "ubuntu" (ver Figura 4.6).

[3] *Transferencia de archivos de la instancia de Linux mediante WinSCP*

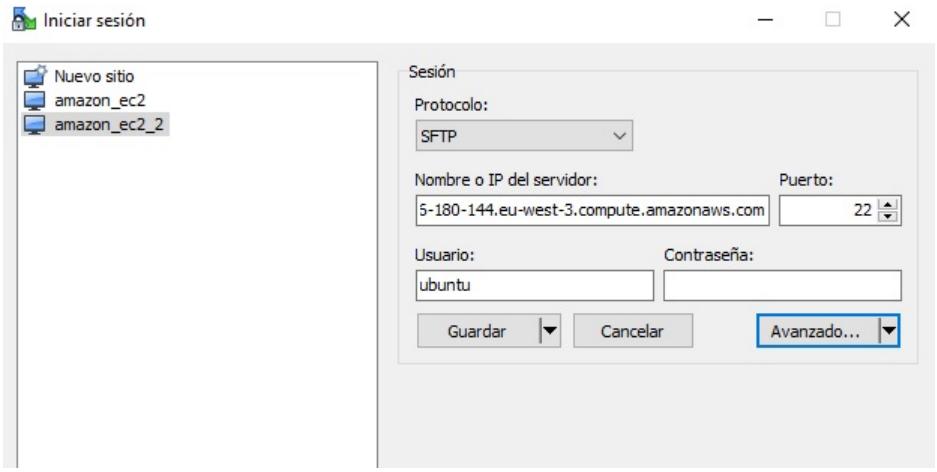

Figura 4.6: Configuración WinSCP

- Acceder a "Avanzado..." y seleccionar, en la navegación de la izquierda, "Autentificación" dentro de "SSH" para importar en "Archivo de clave privada" el fichero de extensión ppk conseguido anteriormente. Pulsar "Aceptar" (ver Figura 4.7).

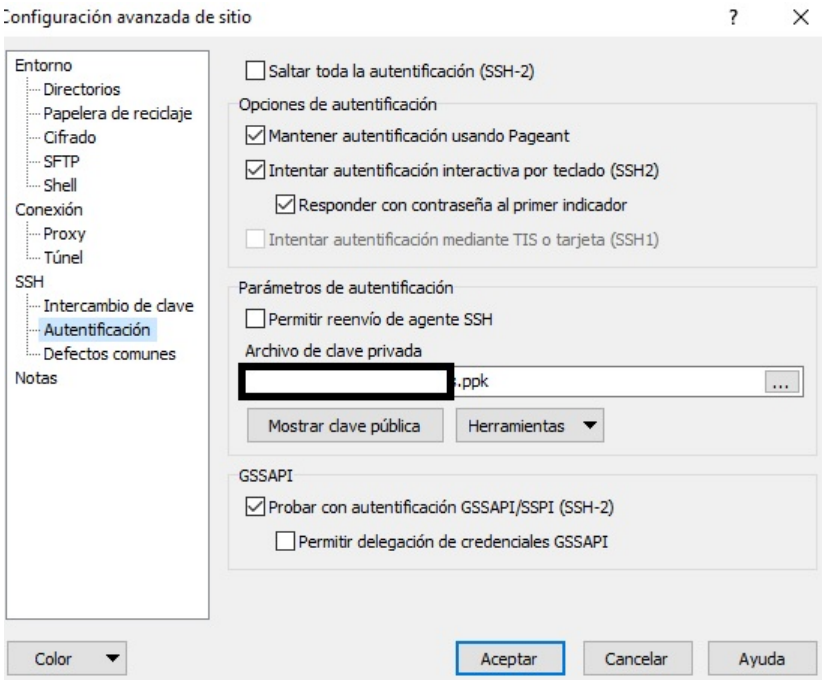

Figura 4.7: Configuración avanzada WinSCP

- Pulsar "Guardar" para que el programa recuerde la configuración y no tener que volver a introducirla.
- Pulsar "Conectar" sobre la configuración recién creada. Si todo va bien será posible acceder al sistema de ficheros de la máquina.

Antes de realizar conexión alguna a la máquina vamos a proceder a poner a punto el correo electrónico de Gmail para utilizarlo en su servidor SMTP en el programa

de notificaciones, generando una contraseña para utilizar únicamente en nuestra aplicación [4].

- Registrarse o iniciar sesión en un correo existente.
- Acceder a la gestión de nuestra cuenta de Google aquí.
- Acceder al apartado "Seguridad". Activar la verificación en dos pasos.
- Tras activar y configurar la verificación en dos pasos, volver al apartado de "Seguridad" y seleccionar "Contraseñas de aplicaciones".
- Seleccionar "otra" aplicación e introducir "Python" u otro nombre que se desee. Presionar en "Generar".
- Guardar la contraseña de aplicación generada. Se utilizará inmediatamente después.

Una vez podemos acceder a la máquina mediante SFTP y SSH, podemos proceder a utilizar PuTTY para realizar las instalaciones de librerías necesarias.

- Ejecutar "sudo apt update" para actualizar el gestor de paquetes.
- Ejecutar "sudo apt install python3-pip" para poder descargar módulos y librerías de Python 3 con la herramienta pip3.
- Ejecutar "pip3 install firebase-admin" para poder operar sobre Firebase. Es posible que tarde un poco.
- Ejecutar "pip3 install bs4", que es la librería Beautiful Soup 4 para tratar código HTML.

Tras dejar preparado lo anterior, es posible acceder mediante SFTP con WinSCP para transferir los archivos que permiten la ejecución del proceso remoto:

- Descargar el fichero "script.py" de nuestro repositorio [5] y obtener las credenciales del proyecto Firebase que se creará posteriormente.
- Descargar el fichero "credenciales.py" de nuestro repositorio y rellenar el contenido de las variables con la ruta a las credenciales de Firebase cuando se generen al crear el proyecto, el usuario SMTP de Gmail y la contraseña de aplicación obtenidas anteriormente.
- Transferirlos a la máquina mediante WinSCP. Es importante que ambos ficheros estén en el mismo directorio y al mismo nivel.
- Para configurar Crontab se debe utilizar el comando "crontab -e". Si es la primera vez que se ejecuta la herramienta nos dejará elegir un editor de texto. Se ha escogido "nano" por su sencillez y ser amigable.
- Añadir una línea nueva tal y como se indica en la Figura 4.8 para que se ejecute todos los días a las 3:00 AM el programa. También es posible añadir otra línea para comprobar que, efectivamente, se ejecuta correctamente Crontab en ese momento (ver Figura 4.9).

[4] Más información en https://support.google.com/mail/answer/185833
[5] https://github.com/Sergidia/PcComponentesTFG_WebScrap

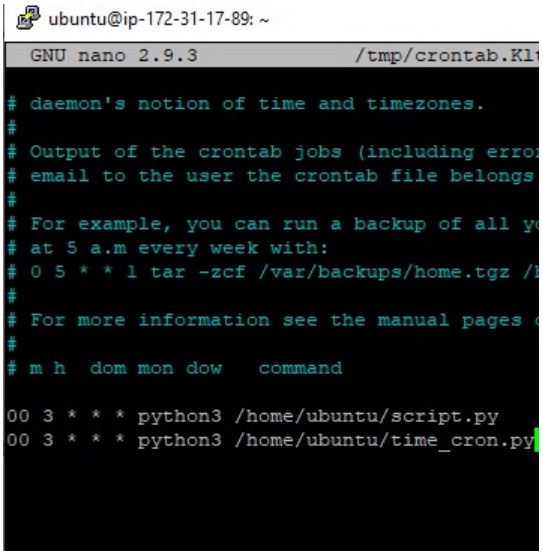

Figura 4.8: Configuración de Crontab

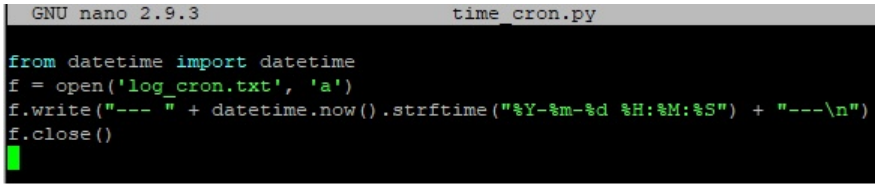

Figura 4.9: Código de time_cron.py

Paso 3 - Creación de un proyecto en Firebase: Para el registro y autenticación de usuarios, además de otras funcionalidades como la de Google Analytics, se creó un proyecto en Firebase que se sincronizaría con el desarrollo en Android Studio.

Para ello es necesario acceder a la web de Firebase[6] e iniciar sesión con una cuenta de Google, como puede ser la cuenta personal de la universidad.

Una vez autenticados creamos un nuevo proyecto pulsando "Crear proyecto", tal y como se puede apreciar en la Figura 4.10, y completamos el formulario, donde indicaremos datos como el nombre del proyecto y la ubicación.

[6]https://console.firebase.google.com/

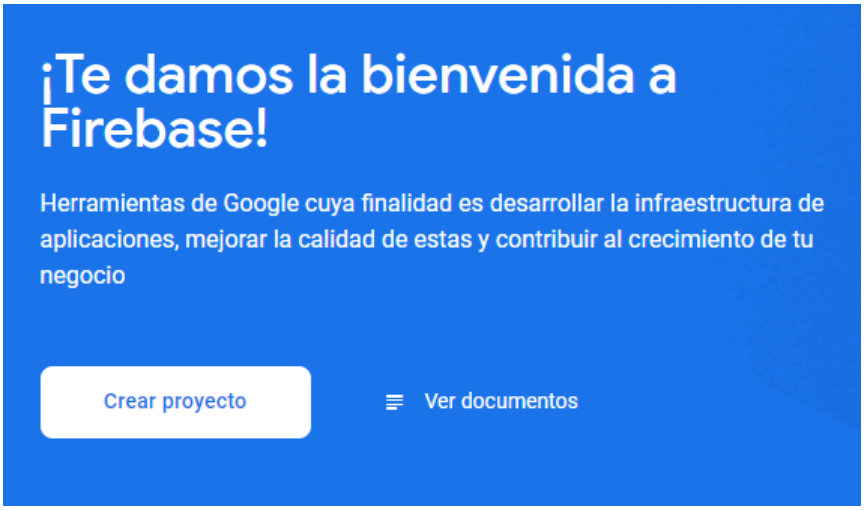

Figura 4.10: Creación del proyecto

Cuando hayamos creado el proyecto nos encontraremos en su página principal (Figura 4.11), donde podremos acceder a los diferentes menús de desarrollo, calidad, analytics y crecimiento. Pulsaremos en "Añadir aplicación"para vincular nuestra aplicación de Android Studio con Firebase. Elegiremos la plataforma sobre la que vamos a desarrollar (Android, iOS, Web o Unity)

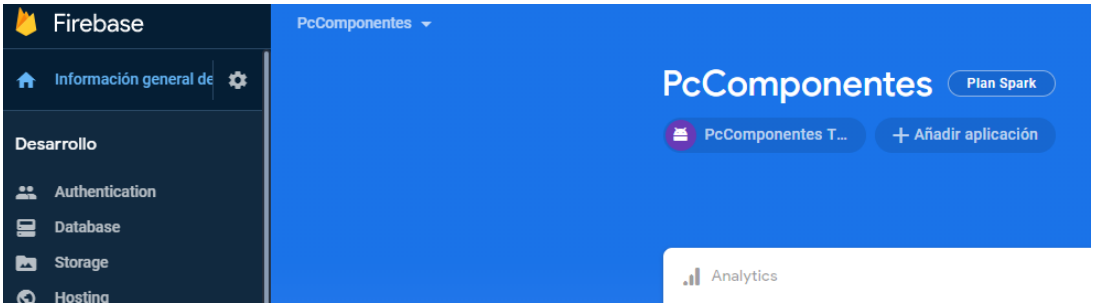

Figura 4.11: Vincular una aplicación

Completaremos de nuevo el formulario, donde nos piden copiar fragmentos del código de nuestra aplicación y pegar en su raíz un JSON generado por Firebase, el cual servirá para conectar la aplicación con la base de datos.

Una vez finalizada la conexión aplicación - Firebase podremos hacer uso de las utilidades de las que dispone, como el acceso a la base de datos de usuarios, llamada "Authentication", y la base de datos no relacional que se encuentra en la pestaña "Database".

Paso 4 - Instalación de bibliotecas: Para poder usar las funcionalidades que requería el proyecto, se añadieron las siguientes librerías:

- Jetbrains
- Firebase
- RecyclerView

- Picasso
- Volley
- Apache

Una vez configurado el entorno se comenzó con el desarrollo de la aplicación.

4.4. Fase de Diseño

Durante el diseño se crearon los diferentes maquetados descritos en la Sección 3.5 para crear las diferentes vistas de la aplicación

4.4.1. Registro de usuarios

En la vista del registro se muestra un formulario donde se solicita un correo electrónico y una contraseña válidas. Adicionalmente, se le pide al nuevo usuario que indique si desea recibir notificaciones por correo o por el móvil cuando haya reducciones del precio de los artículos que añadirá a la lista de seguimiento (Figura 4.12).

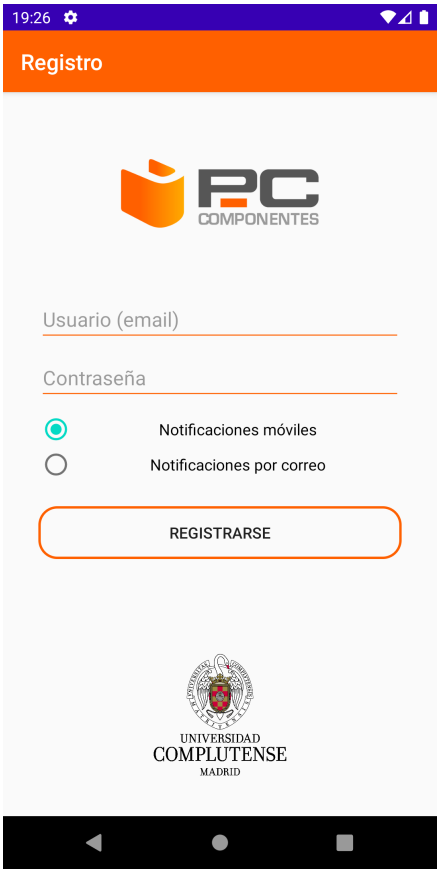

Figura 4.12: Registro

Una vez que se haya confirmado el registro, el usuario deberá verificarlo desde el email que le llegará a su cuenta de correo como en la Figura 4.13.

Figura 4.13: Email de verificación

Si el registro no se ha realizado correctamente (porque el usuario ya exista o el email y/o contraseña no cumplan los requisitos mínimos) se le mostrará un mensaje indicando el error.

4.4.2. Autenticación de usuarios

En la vista del login, correspondiente a la Figura 4.14, se le pide al usuario el correo electrónico y la contraseña.

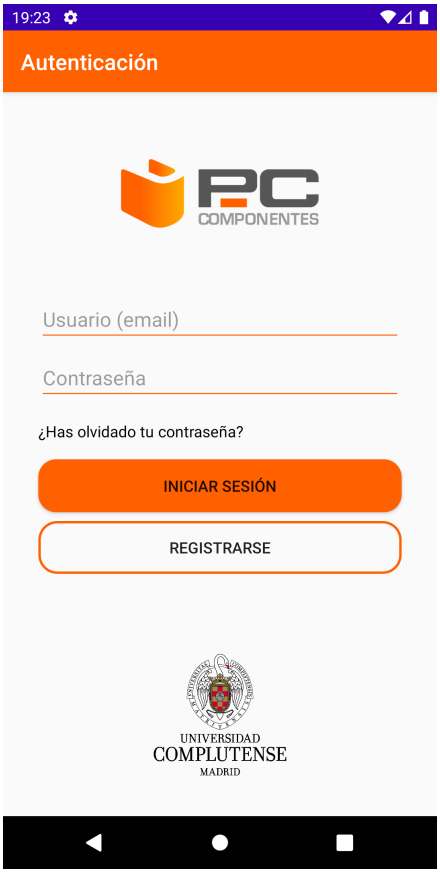

Figura 4.14: Autenticación

Si el usuario no recuerda la contraseña puede pulsar el texto "¿Has olvidado tu contraseña?", que mandará un email a su correo electrónico con un enlace para cambiarla, tal y como se representa en la Figura 4.15.

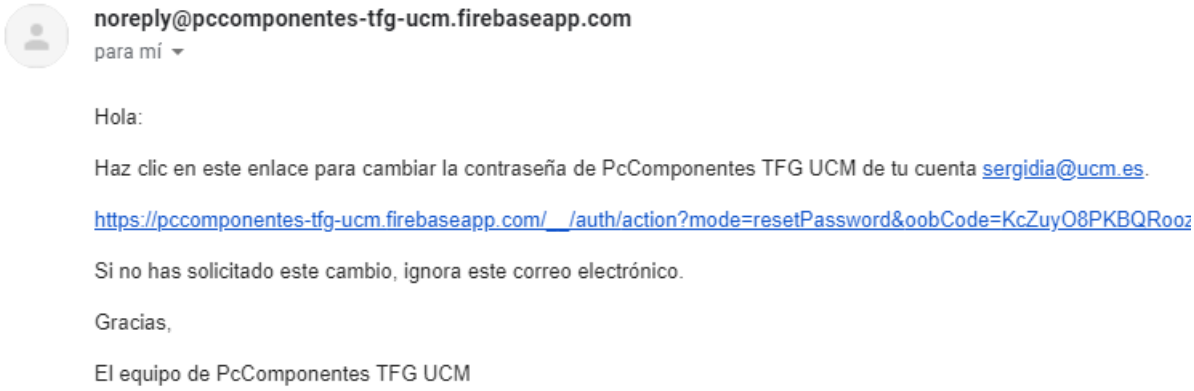

Figura 4.15: Cambiar contraseña

Si la autenticación es satisfactoria se comprueba si el usuario ha verificado su email después del registro. Si no ha sido así, se le muestra un mensaje de error y automáticamente

se le reenvía el correo de verificación. Si la autenticación es correcta y el usuario está verificado se le redigirá a la lista de componentes, de lo contrario se le mostrará un mensaje de error indicando el motivo

4.4.3. Perfil de usuarios

En esta vista el usuario podrá modificar su contraseña y las preferencias de notificación, además de poder eliminar su cuenta de forma permanente, tal y como se ve en la Figura 4.16. Para acceder a ella el usuario tendrá que pulsar en el botón "Perfil" del menú una vez esté autenticado en la aplicación.

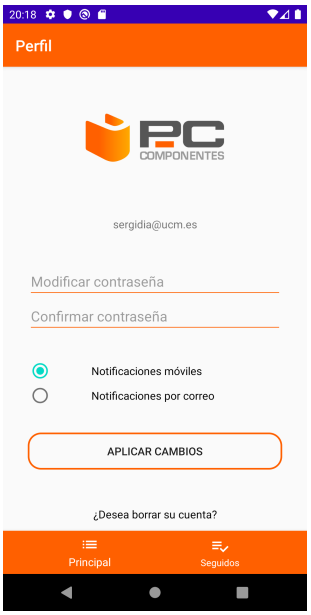

Figura 4.16: Perfil

4.4.4. Lista de componentes

Esta parte de la aplicación consta de dos vistas principalmente. En la vista principal (Figura 4.17), podemos observar una lista con la imagen, nombre, precio y tipo de componente de la página de PcComponentes. La lista carga de forma inicial 10 componentes y haciendo scroll hacia abajo se van cargando componentes de 10 en 10. Esta vista posee además un buscador, en el menú superior, para que encontremos fácilmente el componente que nos interesa. Además, mediante el menú de navegación de la parte inferior se podrá elegir entre esta vista y la vista de los componentes que se siguen.

La vista de los componentes que se siguen (Figura 4.18) es exactamente igual que la anterior con la diferencia de que solo muestra los componentes que el usuario ya ha elegido para hacer un seguimiento de su precio. El tener esta vista es de suma importancia para facilitar la navegación y búsqueda de los usuarios, lo que mejorará su experiencia al usar la aplicación.

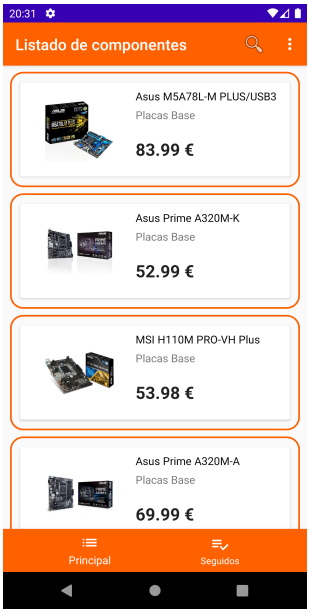

Figura 4.17: Listado de componentes

Figura 4.18: Listado de seguidos

4.4.5. Componente Específico

En esta vista, a la que se accede al seleccionar un componente de la lista, se muestra la información del artículo, un botón que redirige a la propia página de PcComponentes de este producto y una caja de texto donde escribir el precio de notificación si se desea seguir el componente, como se puede ver en la Figura 4.19. Las funcionalidades principales son:

- Añadir un componente a la lista de seguidos con un precio específico.

- Eliminar un componente de la lista de productos seguidos, por lo que el usuario dejará de recibir notificaciones de este.
- Actualizar la información de un componente en base al precio máximo a partir del cual el usuario desee que le avisen.

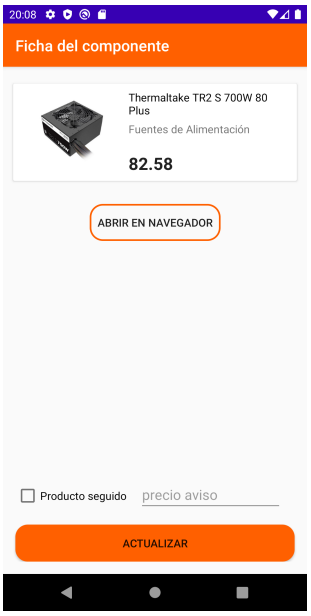

Figura 4.19: Vista del componente

4.5. Fase de Desarrollo

En este apartado explicamos cómo ha sido desarrollado el proyecto, tanto la aplicación móvil como la recogida y notificación de componentes, junto a los problemas presentados en cada apartado.

4.5.1. Recogida y notificación de componentes

Para la realización del *web scraping* es necesario revisar la distribución de páginas de PcComponentes para obtener las URL que interesan, además de analizar el código HTML de las mismas.

Dicho esto, comenzamos por contemplar la página de componentes [7] donde quedan recogidas todas las categorías diferentes que existen en este tipo de artículos (tarjetas gráficas, procesadores...). Observando el HTML, se puede apreciar que cada tipo de componente con su correspondiente redirección está dentro de una etiqueta *div* como en la Figura 4.20, siendo cada enlace una etiqueta *a* con clase *enlace-secundario*. Por ello se recuperarán

[7] https://www.pccomponentes.com/componentes

dichos elementos con atributo *class="enlace-secundario"* y se accederá a cada uno de ellos tal y como se muestra en la Figura 4.21. De este modo, incluso si aparecen nuevas categorías, todas serán accedidas siempre y cuando no se modifique el HTML.

```
<div style="background-image: url(https://cdn.pccomponentes.com/img/fondos/familias/bg-componentes-msi.jpg)"
    class="listado-todas-categorias listado-categorias-NombreFamilia white-card-movil m-b-3 p-a-2">
    <div class="row m-b-1">
        <div class="col-xs-12"><strong class="h4">COMPONENTES</strong></div>
    </div>
    <div class="row">
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-2">
        <ul>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/placas-base">Placas Base</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/procesadores">Procesadores</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/discos-duros">Discos Duros</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/discos-duros-ssd">Discos Duros
                    SSD</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/tarjetas-graficas">Tarjetas
                    Gráficas</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/memorias-ram">Memoria RAM</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/grabadoras-dvd-blu-ray">Grabadoras
                    DVD/Blu Ray</a></li>
        </ul>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-2">
        <ul>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/multilectores">Multilectores</a>
            </li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/tarjetas-sonido">Tarjetas de
                    Sonido</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/torres">Torres/Cajas/Carcasas</a>
            </li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/ventiladores">Ventilación</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/fuentes-alimentacion">Fuentes
                    Alimentación</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/modding">Modding</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/capturadoras">Edición/Captura
                    Vídeo</a></li>
        </ul>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-2">
        <ul>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/cables-internos-de-pc">Cables
                        Internos PC</a></li>
            <li><a class="enlace-secundario" href="https://www.pccomponentes.com/conectividad">Conectividad</a></li>
        </ul>
        </div>
    </div>
</div>
```

Figura 4.20: Elemento div de categorías

```python
def scraping(firebaseDB):
    # Entramos en la página de PcComponentes y sacamos el html
    req = Request('https://www.pccomponentes.com/componentes',
                  headers={'User-Agent': 'Mozilla/5.0'})
    html = urlopen(req).read()
    soup=BeautifulSoup(html, 'html.parser')

    # Buscamos en el html de la página principal de componentes todos los enlaces
    # que llevan a los tipos de componentes
    catComp = soup.find_all(attrs={"class":"enlace-secundario"})

    # Ponemos como no validos los componentes. Segun se vayan haciendo pasadas, si exister
    # se ponen como validos. Si al final no son validos, quiere decir que es posible
    # que hayan desaparecido asi que se notificaria y se eliminarian de la base de datos


    for d in firebaseDB.collection('componentes').stream():
        firebaseDB.collection('componentes').document(d.id).update({'valida':False})

    # Entramos en cada categoría de componentes (procesadores, gráficas, placas base...)

    scrapingCategoria(catComp, firebaseDB)
```

Figura 4.21: Código python para las categorías

Analizando cada categoría, por ejemplo tarjetas gráficas, encontramos un gran problema: no todos los componentes se cargan desde un inicio, se generan dinámicamente con JavaScript según se va navegando por la página, lo cual afecta directamente a la manera de obtener la información de cada uno, ya que al realizar peticiones desde el código sólo se puede acceder a los presentes inicialmente en el HTML (que son unos pocos).

Utilizando herramientas como "Network" de Google Chrome a la vez que se fuerza la carga dinámica de componentes, encontramos el enlace [8] al que se realizan las consultas AJAX cambiando una serie de argumentos como se ve en la Figura 4.22, con un resultado visible y recuperable fácilmente a través del código que permite acceder a dicha página, con una vista como la que se muestra en la Figura 4.23.

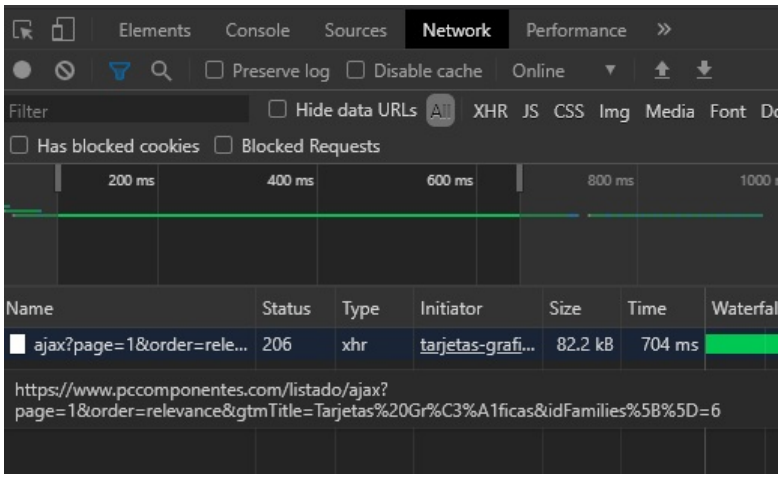

Figura 4.22: Encontrando la petición AJAX

[8]Enlace a la consulta AJAX

Zotac GeForce GTX 1650 Super Twin Fan 4GB GDDR6

179,90 €
Recíbelo **entre el lunes 29 y el martes 30 de junio**
4.57145 Stars
(56)
☐ Comparar ahora
Ver detalle

*P*Promoción *S*Quedan **17** en oferta

PowerColor Red Devil Radeon RX 5700 XT 8GB GDDR6

439,90 €
489,99€
10 %
Recíbelo **entre mañana y el miércoles 3 de junio**
4.54445 Stars
(90)
☐ Comparar ahora
Ver detalle

XFX AMD Radeon RX 590 Fatboy 8GB GDDR5

296,38 €
Recíbelo **entre el jueves 4 y el martes 9 de junio**
4.27025 Stars
(222)
☐ Comparar ahora
Ver detalle

MSI GeForce GTX 1050Ti 4GT OC 4GB GDDR5

146,89 €
Recíbelo **entre mañana y el miércoles 3 de junio**
4.65605 Stars
(580)
☐ Comparar ahora
Ver detalle

Figura 4.23: Resultado de la petición AJAX

Analizando nuevamente el HTML, encontramos que cada componente tiene un formato como el que aparece en la Figura 4.24. Cada uno viene representado por una etiqueta del tipo *article* con toda la información relevante del mismo: nombre, código de artículo, precio y enlace al componente. El único problema es que la imagen es extremadamente pequeña como para ser utilizada adecuadamente, por lo que, si el artículo no existe en nuestra base de datos, se accederá a la página concreta y se obtendrá una con mejor resolución. El hecho de generar tantas peticiones web hace que el proceso pueda tardar hasta dos horas en realizar la recogida de información si no hay nada cargado. En ejecuciones posteriores puede demorarse hasta 20 minutos en función de la cantidad de elementos nuevos cargados por parte de PcComponentes.

En el código se realizan sucesivas modificaciones a los argumentos del enlace AJAX co-

rrespondiente: se incrementa la página hasta que no haya componentes devueltos y se modifica el parámetro "idFamilies" que define la categoría en la que se está consultando. Para obtener el valor correctamente, se debe acceder a cada enlace obtenido en la página global de componentes mencionada anteriormente (Figura 4.20) para obtener una cadena representada por una etiqueta *input* del tipo *hidden* con atributo *data-key* igual a "idFamilies". En la Figura 4.25 queda reflejado el código correspondiente al acceso a cada página dentro de una misma categoría, obteniendo los datos interesantes de cada artículo.

```
<article data-name="Zotac GeForce GTX 1650 Super Twin Fan 4GB GDDR6" data-id="249831" data-price="179.9"
    data-brand="Zotac" data-category="Tarjetas Gráficas" data-stock-web="2" data-list="" data-eec-action="impression"
    data-loop="1" class="tarjeta-articulo expandible">
    <div class="tarjeta-articulo__elementos-basicos">
        <div class="js-article-info" data-id="249831" data-url="/zotac-geforce-gtx-1650-super-twin-fan-4gb-gddr6"></div>
        <div class="tarjeta-articulo__foto">
            <img class="img-fluid lazyload" src="//thumb.pccomponentes.com/w-44-44/articles/24/249831/1.jpg"
                data-src="//thumb.pccomponentes.com/w-220-220/articles/24/249831/1.jpg"
                alt="Zotac GeForce GTX 1650 Super Twin Fan 4GB GDDR6 en PcComponentes">
            <div class="tarjeta-articulo__etiquetas-notificacion">
            </div>
        </div>
        <header class="tarjeta-articulo__nombre">
            <h3>
                <a href="/zotac-geforce-gtx-1650-super-twin-fan-4gb-gddr6" class="GTM-productClick enlace-disimulado"
                    data-name="Zotac GeForce GTX 1650 Super Twin Fan 4GB GDDR6" data-id="249831" data-price="179.9"
                    data-brand="Zotac" data-category="Tarjetas Gráficas" data-stock-web="2" data-list=""
                    data-loop="1">Zotac GeForce GTX 1650 Super Twin Fan 4GB GDDR6</a>
            </h3>
        </header>
        <div class="tarjeta-articulo__precios">
            <div class="tarjeta-articulo__precio-actual">
                179<span class="small">,90 €</span>
            </div>
        </div>
        <div class="tarjeta-articulo__disponibilidad disponibilidad-moderada">
            Recíbelo <strong>entre el lunes 29 y el martes 30 de junio</strong>
        </div>
        <div class="tarjeta-articulo__extras">
            <div class="rating-holder">
                <div class="star-rating rating-xs rating-disabled">
                    <div class="rating-container rating-gly" data-content="NNNNN">
                        <div class="rating-stars" data-content="NNNNN" style="width: 91.429%;"></div>
                    </div>
                    <div class="caption"><span class="label label-primary">4.57145 Stars </span></div>
                </div>
            </div>
            <span class="total_valoration">(56)</span>
        </div>
        <a href="/zotac-geforce-gtx-1650-super-twin-fan-4gb-gddr6" class="GTM-productClick enlace-superpuesto"
            data-name="Zotac GeForce GTX 1650 Super Twin Fan 4GB GDDR6" data-id="249831" data-price="179.9"
            data-brand="Zotac" data-category="Tarjetas Gráficas" data-stock-web="2" data-list="" data-loop="1">
        </a>
        <div class="c-comparator-check-trigger">
            <label class="c-input c-checkbox fat">
                <input type="checkbox" class="js-comparator-check-trigger">
                <span class="c-indicator"></span>
                <span class="text">Comparar<span
                        class="c-comparator-check-trigger__checked-only"> ahora</span></span>
            </label>
        </div>
    </div>
    <div class="tarjeta-articulo__elementos-adicionales">
        <a class="btn btn-block btn-primary" href="/zotac-geforce-gtx-1650-super-twin-fan-4gb-gddr6">
            Ver detalle
        </a>
    </div>
</article>
```

Figura 4.24: Estructura de un componente

```
        while True:
            enlace = "https://www.pccomponentes.com/listado/ajax?" + urlencode(d, quote_via=quote)
            req = Request(enlace, headers={'User-Agent': 'Mozilla/5.0'})
            html = urlopen(req).read()
            soup=BeautifulSoup(html, 'html.parser')

            # Recuperamos cada componente (ver código HTML de la página)
            articulos = soup("article")

            # Si se ha llegado a una pagina que no tiene artículos, dejamos de ver
            if not articulos:
                break

            for articulo in articulos:
                # Para cada componente obtenemos su nombre, código y precio (ver HTML)
                nombreArt = articulo.get("data-name")
                codArt = articulo.get("data-id")
                precioArt = articulo.get("data-price")
                urlComp = "https://pccomponentes.com" + articulo.find(attrs={"GTM-productClick enlace-disimulado"}).get("href")

                ### si no hay componentes cargados tardara bastante!!! (2h aprox aunque puede variar)
                comp = firebaseDB.collection('componentes').document(codArt).get()
                img = ""
                if not comp.exists:
                    req = Request(urlComp, headers={'User-Agent': 'Mozilla/5.0'})
                    html = urlopen(req).read()
                    soup=BeautifulSoup(html, 'html.parser')
                    img = "https:" + soup.find(attrs={"item badgets-layer"})("a")[0].get("href")
                else:
                    img = comp.get('img')
                firebaseDoc = firebaseDB.collection('componentes').document(codArt)
                firebaseDoc.set({
                        'nombre': nombreArt,
                        'precio': float(precioArt),
                        'url': urlComp,
                        'categoria': d['gtmTitle'],
                        'img':img,
                        'valida':True
                        })
            d["page"] += 1
```

Figura 4.25: Código de acceso a componentes de una categoría

Tras obtener la información de cada componente, se almacena en la base de datos. Es interesante destacar que se actualiza el campo "valida" a *true* para tener posteriormente un conocimiento de qué artículos han desaparecido, facilitando de esta manera notificarlo a los usuarios ya sea por correo electrónico o por notificación *push* en función de sus preferencias. De este modo, desaparece todo rastro de aquellos que no han sido recorridos por el proceso de recolección de datos.

Una vez se ha recogido la información del catálogo del día, se procede a notificar a los usuarios interesados en componentes cuyo precio ha bajado del umbral establecido por los mismos o que han sido eliminados. Además, es de suma importancia tener en cuenta la preferencia de ellos, ya sea por correo electrónico o por notificación *push* del móvil.

Entrando en detalles técnicos, la manera de resolver la acción de notificar es recorrer todas las listas de interés de los usuarios dentro de la base de datos, comprobando el precio del componente con el umbral proporcionado y almacenando en un diccionario cierta información de artículos que cumplan las condiciones anteriormente mencionadas, además de eliminar aquellos intereses sobre componentes suprimidos del portal. Se utiliza dicho diccionario para tratar de consumir el menor número de lecturas posible de Firebase que, con su plan gratuito, está limitado (por tanto, si existen muchos usuarios en nuestra aplicación, es muy probable que deje de funcionar a partir de cierta cantidad de uso).

Gracias al uso de Firebase desde Python con el módulo *messaging* de Firebase_admin

se facilita en gran medida el proceso de notificaciones *push* a los dispositivos móviles de los usuarios. Esto es posible gracias a que se genera un *token*, que es un código único para identificar los teléfonos que utilizan nuestra aplicación que va variando cada vez que se inicia sesión. En el caso de informar acerca de componentes eliminados se escribe el nombre de cada uno de ellos como mensaje de la notificación (ya que no deberían ser suprimidos una gran cantidad de ellos habitualmente) pero, en el caso de las ofertas, se indica de manera generalizada que algunos artículos de interés han cumplido la condición de bajar del precio umbral propuesto por el usuario, debido a que puede haber una gran cantidad de ellos y hacer demasiado extenso el mensaje.

También, gracias al uso del servidor SMTP de Gmail, es posible realizar fácilmente el envío de notificaciones por correo electrónico. La estructura de los mensajes enviados consiste en un asunto que indica de qué se trata en el mismo (por ejemplo, que hay artículos que te interesan que han sido rebajados; o componentes que llamaban la atención del usuario que han sido eliminados), con una lista de imágenes y nombres con hipervínculos a la información detallada de cada uno de ellos. En las Figuras 4.26, 4.27 y 4.28 presentadas a continuación se puede comprobar ejemplos de los correos electrónicos y notificaciones *push*.

Figura 4.26: Notificaciones por email. Componentes rebajados.

Figura 4.27: Notificaciones por email. Componentes eliminados.

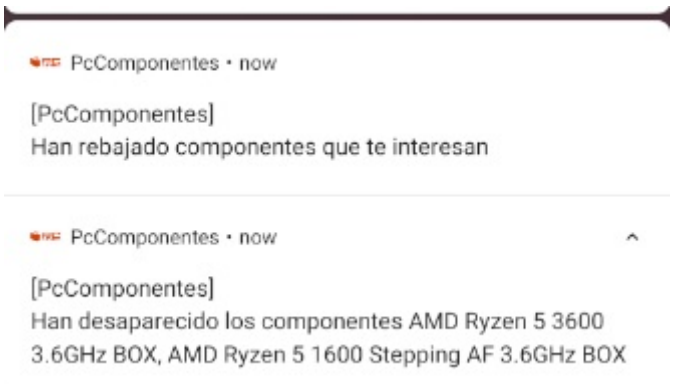

Figura 4.28: Notificaciones push

Cada vez que se ejecuta el proceso de recogida y notificación de componentes, quedan almacenadan en un fichero llamado "timelog.txt" el conjunto de fechas en el que se utiliza el programa asociadas a la duración de cada ejecución para llevar un control de la correcta invocación del código y del tiempo que tarda en finalizar.

También es de suma importancia recalcar el hecho de la gran dependencia que encontramos relacionada con que no haya cambios bruscos en el código HTML de las páginas de PcComponentes, además de tener limitaciones en el número de operaciones de escritura y lectura en Firebase y el número de correos enviados por día a través del servidor SMTP de Gmail. Por otra parte, la capa gratuita de Amazon AWS con la instancia EC2 permite un procesamiento suficiente para pocos usuarios pero, debido a sus especificaciones, no se permite paralelización del código ni el mantenimiento de más instancias del mismo tipo.

De este modo, en la migración de MySQL a Firebase se experimentó un recorte en la cantidad de información a almacenar. Inicialmente, por motivos de auditoría, consideramos importante poseer un registro de fechas de inicio y cierre de sesión, así como de

las notificaciones enviadas. Se descartó esta idea debido a las limitaciones mencionadas anteriormente.

En lo referente a la ejecución automática del proceso descrito anteriormente de recogida y notificación de componentes, se ha optado por utilizarlo una vez al día (a las 3:00 AM en horario de la máquina EC2 de Amazon AWS) debido a las limitaciones que se presentan en el uso de Firebase, Gmail SMTP y la propia instancia EC2: es difícil adaptarlo a un gran número de usuarios ya que ralentizaría los tiempos de ejecución del programa por el incremento de procesamiento y haber mayor número de correos electrónicos a enviar, afectando directamente a las lecturas y escrituras realizadas sobre la base de datos. En la Sección 6 se analizan estas restricciones y cómo podrían ser solventadas.

4.5.2. Aplicación móvil

En esta sección revisaremos el desarrollo de las partes *back-end* y *front-end* definidas en la aplicación móvil.

Back-End

Back-end es la parte de la aplicación que se encarga del procesamiento de datos, que son recibidos o mostrados por el Front-end.

- **Login:** Se hace una petición a Firebase con el usuario y la contraseña introducidos como parámetros, para realizar la autenticación como se muestra en la Figura 4.29.

```
FirebaseAuth.getInstance()
        .signInWithEmailAndPassword(userText.text.toString(),
                passwordText.text.toString()).addOnCompleteListener { it: Task<AuthResult!>

            if (it.isSuccessful){

                // Comprobamos si hace falta actualizar el token almacenado previamente
                CheckData.actualizarToken( email: it.result?.user?.email ?: "")

                // Comprobamos si el usuario ha verificado su registro por email
                if (FirebaseAuth.getInstance().currentUser?.isEmailVerified == true) {

                    // Si el usuario verificó su registro redirigimos al listado de componentes
                    showList()
                }

                // Si no verificó su registro se muestra un pop up con un recordatorio y se le vuelve a enviar el email
                else {
                    showAlert( mensajeError: "No se ha podido iniciar sesión, debe confirmar el registro desde el email recibido en su cuenta de correo")
                    FirebaseAuth.getInstance().currentUser?.sendEmailVerification()
                }

            } else {
                showAlert( mensajeError: "No se ha podido iniciar sesión, compruebe los datos introducidos")
            }
        }
lse showAlert( mensajeError: "La contraseña debe tener mínimo 6 caracteres")
howAlert( mensajeError: "Se debe introducir una cuenta de correo electrónico")
lert( mensajeError: "Los campos de usuario y contraseña no pueden estar vacíos")
```

Figura 4.29: Login

- **Registro:** Se genera una conexión con Firebase para crear un usuario con email y contraseña, que es el tipo de autenticación que hemos implementado, con código como el de la Figura 4.30.

```
FirebaseAuth.getInstance()
        .createUserWithEmailAndPassword(userText.text.toString(),
                passwordText.text.toString()).addOnCompleteListener { it: Task<AuthResult!>
            if (it.isSuccessful){

                // Si los datos son correctos se inicia una instancia en Firestore para almacenar las preferencias de notificación del usuario
                val db = FirebaseFirestore.getInstance()

                // Recuperamos el token del móvil del usuario
                val token: String = MyFirebaseMessagingService.getInstanceToken()

                // Creamos un documento para la colección de usuarios, que contendrá los datos del objeto Notificacion creado posteriormente
                val notificacionesUsuario: DocumentReference = db.collection( collectionPath: "usuarios").document(userText.text.toString())

                // Creamos un objeto de tipo Notification, que contiene los booleanos de los dos tipos de notificaciones según la elección del usuario y el token del
                val notif: NotificationDocumentObject = NotificationDocumentObject(notifPush.isChecked, notifEmail.isChecked, token)

                // Insertamos el documento en la base de datos
                notificacionesUsuario.set(notif).addOnCompleteListener{ it: Task<Void!>
                    if (it.isSuccessful){

                        // Enviamos el email de verificación al correo del usuario
                        FirebaseAuth.getInstance().currentUser?.sendEmailVerification()

                        // Si el registro ha sido satisfactorio redirigimos a la vista del login
                        showMain()
```

Figura 4.30: Registro

- **Perfil:** Se recupera la instancia de Firebase para modificar la contraseña y/o las preferencias de notificación del usuario como se puede contemplar en la Figura 4.31.

```
FirebaseAuth.getInstance().getCurrentUser()
        ?.updatePassword(passwordText.text.toString())
            ?.addOnCompleteListener { it: Task<Void!>
                if (it.isSuccessful){

                    // Si los datos son correctos se inicia una instancia en Firestore para almacenar las preferencias de notificación del usuario
                    val db = FirebaseFirestore.getInstance()

                    // Recuperamos el token del móvil del usuario
                    val token: String = MyFirebaseMessagingService.getInstanceToken()

                    // Accedemos al documento del usuario
                    val notificacionesUsuario: DocumentReference = db.collection( collectionPath: "usuarios").document(userText.text.toString())

                    // Creamos un objeto de tipo Notification, que contiene los booleanos de los dos tipos de notificaciones según la elección del usuario y el token
                    val notif: NotificationDocumentObject = NotificationDocumentObject(notifPush.isChecked, notifEmail.isChecked, token)

                    // Insertamos el documento en la base de datos
                    notificacionesUsuario.set(notif).addOnCompleteListener{ it: Task<Void!>
                        if (it.isSuccessful){
                            // Si se ha actualizado correctamente redirigimos al listado de componentes
                            showList()
```

Figura 4.31: Perfil

- **Token:** Para poder enviar las notificaciones *push* necesitamos almacenar el token del dispositivo móvil del usuario. Los token pueden cambiar si se reinstala la aplicación o el usuario se autentica en un dispositivo distinto. En las Figuras 4.32 y 4.33 se pueden observar fragmentos de código que permiten recuperar el token y actualizarlo en la base de datos.

```
companion object {
    private const val TAG = "Notificaciones Push"
    private var tokenData = ""

    /**
     * Se recupera el token del dispositivo del usuario
     */
    fun getInstanceToken(): String {

        if (tokenData == "") {

            var usuario = FirebaseInstanceId.getInstance().instanceId.addOnSuccessListener { it: InstanceIdResult!
                this.tokenData = it.token
                Log.d(TAG,  msg: "Token actualizado: $tokenData")
            }
        }
        return tokenData
    }
}
```

Figura 4.32: Obtener token

```
/**
 * Actualizamos el registro del token de la base de datos
 */
fun actualizarToken(email: String) {

    // Recuperamos el token del móvil del usuario
    val token: String = MyFirebaseMessagingService.getInstanceToken()

    // Obtenemos la instancia de Firebase
    val db = FirebaseFirestore.getInstance()

    // Recuperamos el documento del usuario
    val notificacionesUsuario: DocumentReference = db.collection( collectionPath: "usuarios").document(email)

    // Convertimos el documento recuperado en un objeto de tipo NotificationDocumentObject
    notificacionesUsuario.get().addOnSuccessListener { documento ->
        val notificaciones = documento.toObject(NotificationDocumentObject::class.java)

        if (notificaciones != null && token != "") {

            // Si el token recuperado del documento no es igual al token actual lo actualizamos y subimos el documento a la base de datos
            if (notificaciones.getToken() != token) {

                notificaciones.setToken(token)
                notificacionesUsuario.set(notificaciones)
            }

        }
    }
```

Figura 4.33: Actualizar token

- **Usuario autenticado:** En cualquiera de las vistas de la aplicación se comprueba si el usuario sigue autenticado y, de lo contrario, se le redirige a la vista del login. Además, en la vista del login, también se comprueba si el token ha sido modificado. En la Figura 4.34 se observa la parte del código que permite esta funcionalidad.

```
// Comprobamos si el usuario está autenticado
val email: String = FirebaseAuth.getInstance().currentUser?.email.toString()

// Si lo está recuperamos el token y comprobamos si debemos actualizarlo. Luego redirigimos al listado de componentes
if (email != "" && email != "null") {
    // Comprobamos el token y lo actualizamos de hacer falta
    CheckData.actualizarToken(email)

    // Independientemente de que hayamos o no actualizado el token, redirigimos a la vista del listado de componentes si verificó su registro
    if (FirebaseAuth.getInstance().currentUser?.isEmailVerified == true)
        showList()
}
// Si no está autenticado o verificado generamos la vista del login
 else iniciarMain()
```

Figura 4.34: Comprobar usuario autenticado

- **Listado de artículos:** es necesario recuperar los componentes de la base de datos, por lo que se accede a Firebase y se obtienen todos los diez primeros documentos de la colección “componentes”, como se muestra en la Figura 4.35 y Figura 4.36.

```
/**
 * Inicialización de la lista de componentes recuperando los 10 primeros de la base de datos
 */
private void iniciarListaComponentes() {

    Log.d( tag: "Listado Main",  msg: "Se inicializa la lista de componentes");

    db.collection( collectionPath: "componentes") CollectionReference
            .limit(10) Query
            .get() Task<QuerySnapshot>
            .addOnSuccessListener((OnSuccessListener) (queryDocumentSnapshots) → {
                    for (QueryDocumentSnapshot document : queryDocumentSnapshots) {

                        addComponente(document);
                    }
                    cargarRecyclerView();
                    isLoading = false;
            });
}
```

Figura 4.35: Recogida en la aplicación de los componentes

```
/**
 * Añade un componente a la lista de componentes
 *
 * @param document documento del componente
 */
public void addComponente(QueryDocumentSnapshot document) {

    Log.d( tag: "Componente Main",  msg: "Se añade al a lista un documento con id: " + document.getId());

    Map<String,Object> componente = document.getData();
    Item component = new Item(
            document.getId(),
            (String)componente.get("nombre"),
            (String)componente.get("img"),
            ((Number)componente.get("precio")).doubleValue(),
            (String)componente.get("url"),
            (String)componente.get("categoria"),
            (Boolean)componente.get("valida"));

    listadoComponentes.add(component);
    setLastVisible(document);

    Log.d( tag: "Componente Main",  msg: "El documento se corresponde con el componente con nombre: " + componente.get("nombre"));
}
```

Figura 4.36: Añadir componentes a la lista

Estos documentos tienen un identificador único, que es el ID dado por la web de PcComponentes, y contienen los datos de cada artículo, como la categoría, la url de la imagen, su precio, etc. con una estructura similar a la detallada en la descripción de la base de datos. En la Figura 4.37 se puede comprobar cómo queda almacenada la información en Firebase.

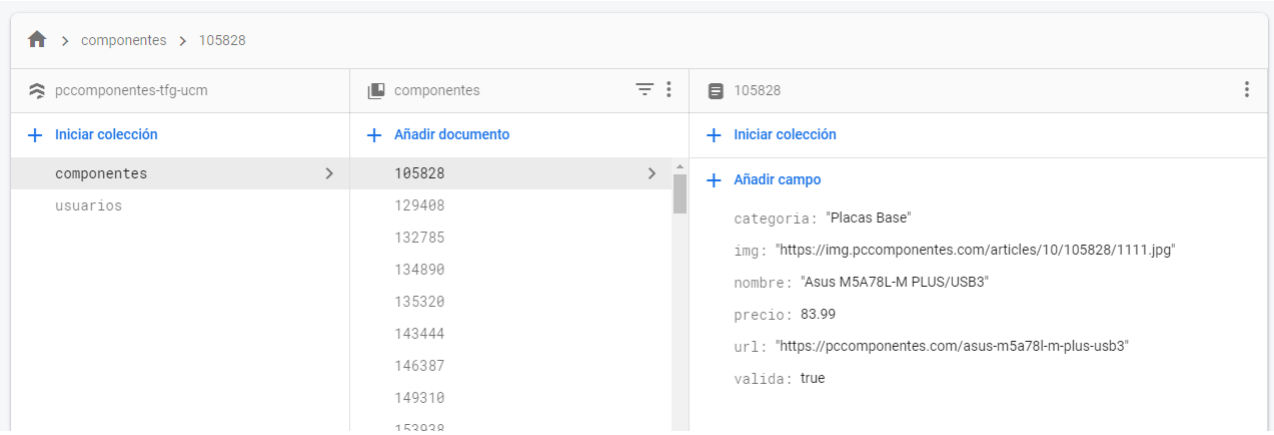

Figura 4.37: Lista de componentes en Firebase

- **Listado de componentes seguidos:** se realiza otro acceso a Firebase para conseguir la lista de intereses del usuario. Tras recuperar la lista de interés se muestra cada componente, de un modo similar a lo mostrado en la vista del listado de artículos (Figura 4.38 y Figura 4.39).

```
/**
 * Inicialización de la lista de componentes recuperando los 10 primeros seguidos de la base de datos
 */
private void iniciarListaComponentes() {

    Log.d( tag: "Listado Seguidos", msg: "Se inicializa la lista de componentes");

    // Se recuperan los 10 primeros documentos de la lista de interés del usuario
    db.collection( collectionPath: "usuarios") CollectionReference
            .document(email) DocumentReference
            .collection( collectionPath: "interes") CollectionReference
            .limit(10) Query
            .get() Task<QuerySnapshot>
            .addOnSuccessListener((OnSuccessListener) (queryDocumentSnapshots) → {

                // Una vez recuperados se itera sobre el resultado de la query. Cada documento recuperado se corresponde con un
                // por lo que se busca el documento de tipo 'componente' que corresponda y se añade a la lista de componentes.
                // Además, guardamos el documento de 'interés' en la variable "lastVisibleSeguidos"
                for (QueryDocumentSnapshot document : queryDocumentSnapshots) {

                    addComponente(recuperarComponente(document));
                    setLastVisibleSeguido(document);
                }

                cargarRecyclerView();
                isLoading = false;
            });
}
```

Figura 4.38: Recogida de la lista de seguimiento de un usuario

```java
/**
 * Método que devuelve el documento de la colección 'componentes' que tenga el mismo nombre que el documento de 'interés'
 *
 * @param document Documento de interés del usuario
 * @return Documento del componente relacionado con ese interés
 */
private Task<DocumentSnapshot> recuperarComponente(QueryDocumentSnapshot document) {
    Task<DocumentSnapshot> task = db.collection( collectionPath: "componentes")
            .document(document.getId())
            .get();

    return task;
}
```

Figura 4.39: Recuperar componente de la base de datos

La lista de interés del usuario es una colección, llamada "interés", que está dentro del documento de cada usuario. Esta colección almacena documentos, que se corresponden con cada artículo seguido, y que contienen el precio a partir del cual quiere ser notificado el usuario. Ambas estructuras quedan reflejadas en las Figuras 4.40 y 4.41.

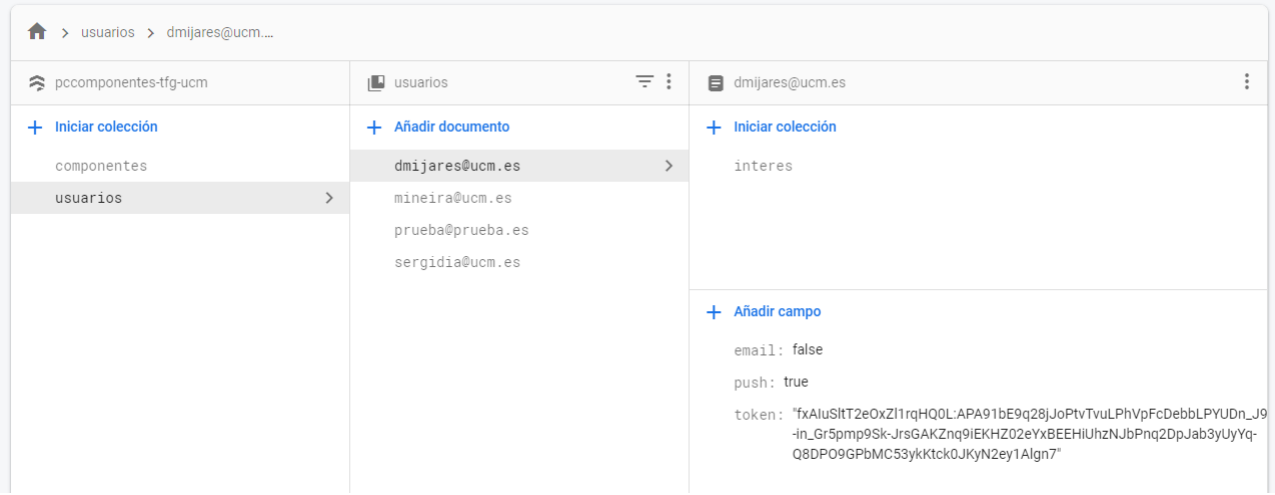

Figura 4.40: Información de un usuario en Firebase

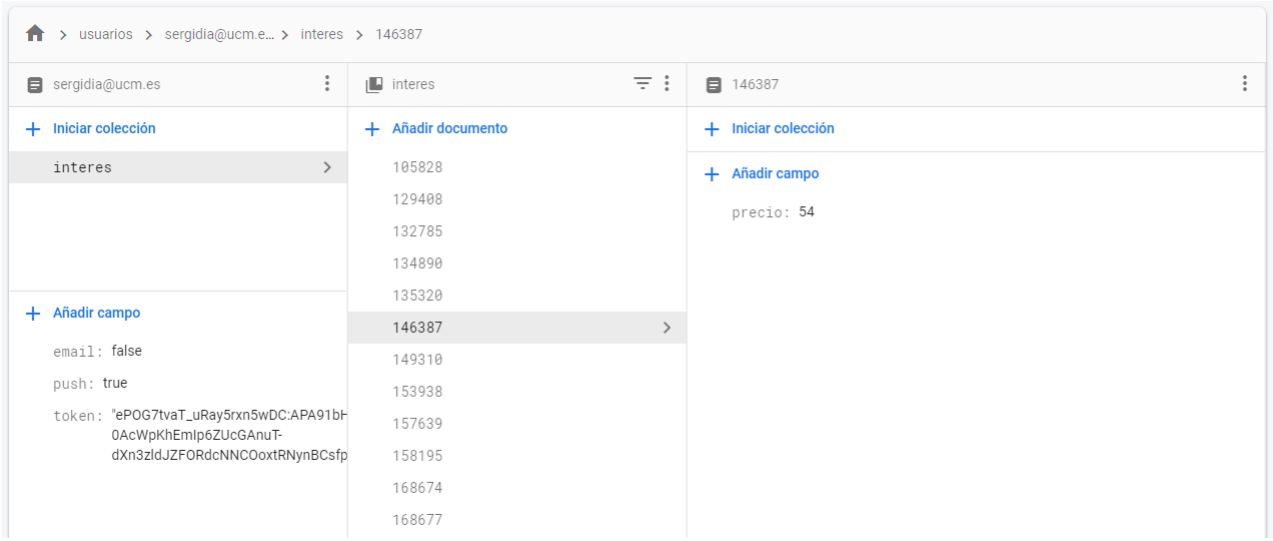

Figura 4.41: Lista de intereses de un usuario en Firebase

- **Ficha de un componente especifico:** el método "actualizarNotificacion()" comprueba en qué estado se encuentra el producto, añadiéndolo a la lista de seguimientos de quererlo el usuario, modificando su precio si ya estaba añadido o eliminándolo de la lista si ya no le interesa, actualizando en cualquiera de los casos la base de datos de Firebase con código como el que queda representado en la Figura 4.42.

```
if(seguir.isChecked()){

    if (precioNoti.getText().toString().isEmpty()) {
        showAlert( mensajeError: "Debe indicar el precio a partir del cual ser notificado para seguir un componente");
    }
    else {
        try {

            // Si no estaba seguido se crea un seguimiento y si lo estaba pero el precio de seguimiento ha cambiado se actualiza
            if(!seguido || seguimiento.getPrecioMax() != Double.parseDouble(precioNoti.getText().toString())){

                Map<String,Object> aux = new HashMap<>();
                aux.put("precio",Double.parseDouble(precioNoti.getText().toString()));

                db.collection( collectionPath: "usuarios").document(email) DocumentReference
                        .collection( collectionPath: "interes").document(seguimiento.getCodigo())
                        .set(aux) Task<Void>
                        .addOnSuccessListener((OnSuccessListener) (aVoid) → {
                                seguimiento.setPrecioMax(Double.parseDouble(precioNoti.getText().toString()));
                                if(!seguido){
                                    seguido = true;
                                    Toast.makeText(getApplicationContext(), text: "Ahora sigues este componente",Toast.LENGTH_SHORT).show();
                                }
                                else{
                                    Toast.makeText(getApplicationContext(), text: "Precio de seguimiento actualizado",Toast.LENGTH_SHORT).show();
                                }
                        })
                        .addOnFailureListener((e) → {
                                showAlert( mensajeError: "Error al seguir el componente");
                        });
            }
        }
        catch (NumberFormatException nE) {
            showAlert( mensajeError: "Debe introducir un número en el campo del precio");
        }
    }
}
else{

    // Si ya estaba seguido se deja de seguir
    if(seguido){
        db.collection( collectionPath: "usuarios").document(email) DocumentReference
```

Figura 4.42: Seguimiento del producto por un usuario

Front-End

Front-end es la parte de la aplicación enfocada a representar la información al usuario, ya sean botones con los que puede interactuar o datos que puede introducir o leer, como el login o las fichas de los componentes, respectivamente.

- **Login:** Consta de un formulario con los campos usuario y contraseña, además de los botones "¿Has olvidado tu contraseña?", "Iniciar sesión' y "Registrarse". El botón "¿Has olvidado tu contraseña?" enviaría una solicitud a Firebase para que el usuario reciba un correo electrónico que contenga una url para reiniciar la contraseña.
- **Registro:** Contiene un formulario con los campos usuario y contraseña, además de las preferencias de notificación, y un botón "Registrarse".
- **Listado de componentes:** Muestra las fichas de los componentes guardados en la base de datos, cargados en bloques de 10 para evitar ralentizaciones en móviles con menores prestaciones. Esta vista junto a la de listado de seguidos utiliza a su vez

la barra superior que da acceso al buscador y las opciones para ver nuestro perfil o cerrar sesión, y otro, para el *bottom navigation bar* que nos permite desplazarnos entre la lista de todos los productos y los seguidos, tal y como se puede comprobar en la Figura 4.43 presentada a continuación.

Figura 4.43: Barra superior vista componentes y seguidos

Además, se necesita otro xml para definir la estructura y diseño de cada uno de los productos de la lista. Este xml está formado por varios LinearLayout que van conteniendo en su interior a un imageView para la imagen del producto y varios textView para el nombre, precio y clase de producto. El plano queda reflejado en la Figura 4.44

Figura 4.44: Plano de estructura de un item del recycler

- **Listado de seguidos:** Muestra las fichas de los componentes seguidos por el usuario, cargados en bloques de 10.
- **Ficha del componente:** Muestra la información específica de un componente, siendo esta el nombre, la imagen, la categoría y el precio. Además, permite seleccionar la ficha para que se abra una ventana del navegador con la url de ese artículo en PcComponentes. También, permite empezar a seguir un producto, dejar de seguirlo y actualizar el precio de seguimiento.
- **Perfil:** Contiene campos para actualizar la contraseña, las preferencias de notificación y borrar la cuenta de forma permanente.

Capítulo 5

Contribuciones al Proyecto

En este capítulo se presentan las aportaciones realizadas por cada integrante en el proyecto.

5.1. Miguel Neira Martín

Participación en la elaboración de la memoria.

Investigar e intentar diferentes alternativas para realizar *web scraping*, optando finalmente por Beautiful Soup 4 de Python 3 por su sencillez.

Prueba, despliegue y mantenimiento de las máquinas EC2, para ejecutar el proceso automático y montar el servidor web API para la aplicación móvil, y RDS de Amazon AWS para la base de datos de componentes, interés y notificaciones en MySQL antes de migrar a Firebase.

Adaptación del proceso automático para utilizar Firebase como base de datos a medida que se fue avanzando en el proyecto, para así permitir una integración más fácil con la aplicación móvil.

Implementación y desarrollo del proceso automático de recogida de componentes y envío de notificaciones por correo electrónico y *push* si se producen rebajas o se eliminan artículos de interés para los usuarios.

Proporcionar y analizar ideas y problemas a lo largo del proyecto para el mejor desarrollo del mismo.

Probar y retocar parte del código de la aplicación móvil.

5.2. Sergio Díaz Renedo

Participación en la elaboración de la memoria.

Creación y configuración del proyecto en Firebase, además de los repositorios de GitHub usados para llevar un control de versiones de la parte de web scraping y de la aplicación móvil.

Desarrollo inicial de la aplicación móvil, incluida la autenticación, registro y comunicación con Firebase. Colaboración en la implementación de los listados de componentes y en la comunicación de Firebase con el script de web scraping.

Adaptación del proyecto para soportar código en Java y Kotlin al mismo tiempo.

5.3. Darío Mijares Muñiz

Participación en la elaboración de la memoria.

Creación de las vistas de los mock-ups orientativos de la interfaz de la aplicación móvil en las vistas de inicio de sesión, registro de usuario y la vista principal.

Desarrollo de la aplicación móvil, incluyendo la comunicación entre la parte de los componentes con Firebase para activar, eliminar, y actualizar seguimiento de productos. Además, un buscador para facilitar al usuario que encuentre un producto específico.

Trabajado en el testeo y comprobación del correcto funcionamiento de la aplicación.

Capítulo 6

Trabajo Futuro

Analizando el estado actual del proyecto, se pueden proponer varias ideas a desarrollar en un futuro que se van a tratar en las próximas secciones.

6.1. Integración con PcComponentes

Una posible idea es la integración con PcComponentes, permitiendo una vinculación con la cuenta del portal y así realizar operaciones adicionales que se permitan.

6.2. Limitaciones con Firebase

Si se desea llegar a muchos usuarios y publicar la aplicación, necesitamos dar soporte a que se pueda utilizar sin restricciones o se dejará de mostrar interés por el proyecto. Para ello, utilizar un plan de pago por uso es una de las posibles opciones, intentando también adaptar el código para ahorrar el mayor número de lecturas y escrituras posible. De este modo, sería posible una mejora en la precisión de las notificaciones debido a que podría ser interesante ejecutar la recogida de información varias veces al día y así no perder ofertas *flash* de duración muy limitada en la tienda.

6.3. Limitaciones con Amazon AWS

Pasados los 12 meses de prueba de la capa gratuita sería interesante, en el caso de seguir funcionando la aplicación, adaptarse al pago por uso que proporciona Amazon AWS para así permitir una mejora en el código de recogida y notificación de componentes a través de la paralelización del mismo, ahorrando así tiempos de ejecución.

6.4. Mejoras en el código Python

Aunque ya se ha comentado en secciones anteriores, sería posible mejorar, dentro de las posibilidades, la paralelización y el número de escrituras y lecturas que se realizan contra la base de datos.

6.5. Añadir publicidad

Implementar las mejoras propuestas anteriormente hace que haya costes presentes. Para tratar de mantener los servicios con el menor gasto, si es posible, podría tratar de añadirse publicidad poco abusiva a la aplicación (que haya muchos anuncios que además dificulten el uso del programa haría que los usuarios dejaran de mostrar interés en nuestro proyecto).

6.6. Programa afiliados PcComponentes

Inscribirse al programa de afiliados de PcComponentes [26] nos permitiría ganar un porcentaje de comisión por cada venta que generáramos, lo que nos podría permitir mantener los costes de la aplicación sin necesidad de añadir publicidad.

6.7. Mejoras en SMTP

Utilizar servidores SMTP que no presenten limitaciones de envíos diarios como en el caso de Gmail, buscando también un diseño más fácil de ver por parte de los usuarios en los correos enviados.

6.8. Ampliación hacia otros artículos que no sean componentes de ordenador

PcComponentes es una tienda que oferta mucha variedad de artículos más allá de componentes. Un posible trabajo futuro podría ser extender la funcionalidad a recoger más tipos de artículos.

6.9. Publicar la aplicación

Otra posible idea sería la publicación de la aplicación para que todo el mundo pueda utilizarla, por ejemplo, a través de la tienda oficial de Google, Play Store [14]. En un futuro, si así se desea, será interesante comprobar los requisitos y someter el proyecto a los procesos necesarios para poder ser subido al servicio.

6.10. Mejoras en la búsqueda de componentes

Una idea interesante sería la de implementar filtros avanzados para permitir diferentes tipos de búsqueda en las listas de componentes. Habría que estudiar qué utilidades proporciona Firebase al respecto o investigar si, una vez cargada la lista, hay posibilidades de facilitar dicha tarea.

6.11. Tema oscuro

Para satisfacer la demanda tan popular del denominado *Dark Theme* [34], que consiste en el uso de colores oscuros y negros con el objetivo de reducir el consumo de batería y la fatiga ocular.

Capítulo 7

Conclusiones

En primer lugar, en lo referente al *web scraping* de PcComponentes, gracias a distintas mejoras en el código se consiguió pasar de que cada ejecución durara alrededor de dos horas a que solo durase ese tiempo la primera vez, mientras que las sucesivas ejecuciones tuvieran una duración inferior a diez minutos.

Por otra parte, tras el proceso de diseño y e inicio del desarrollo de la aplicación, nos dimos cuenta de que el cambio de la base de datos a Firebase nos facilitaría mucho su implementación. Tras codificar la autenticación y construir la base de datos conseguimos que la aplicación funcionara de una forma mas eficiente y segura. Además, nos facilitó la generación de notificaciones push.

En referencia al diseño de la interfaz de la aplicación, la creación de mock-ups ayudó al equipo a tener una idea general de como debía ser su aspecto general para poder garantizar una mejor experiencia de usuario.

En definitiva, a lo largo del desarrollo de las distintas partes del proyecto el equipo ha ido identificando posibles mejoras tanto a nivel de código como a nivel de utilidad de las distintas plataformas usadas. Todo ello, con el objetivo de conseguir que el resultado final fuera el mas óptimo posible.

Capítulo 8

Conclusions

Firstly, with regard to web scraping PcComponentes, thanks to different code improvements it was possible to reduce the execution time of the script, which initially was two hours each, to less than ten minutes each successive execution after the first one.

On the other hand, after the design process and the start of the development, we realized that changing the database to Firebase would make the implementation much easier for us. After coding the authentication and building the database, we managed to make the application work in a more efficient and secure way. In addition, it simplified the implementation of push notifications.

In reference to the design of the application interface, mock-ups helped the team to have a general idea of the expected appearance in order to guarantee a better user experience.

In conclussion, throughout the development of the different parts of the project the team has been identifying code and utility improvements, aiming to achieve the most optimal result.

Índice de figuras

Bibliografía

[1] Anaconda Python/R Distribution - Free Download. https://www.anaconda.com/distribution/.

[2] AWS | Cloud Computing - Servicios de informática en la nube. https://aws.amazon.com/es/.

[3] Beautiful Soup Documentation – Beautiful Soup 4.4.0 documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[4] Bienvenidos a Steam. https://store.steampowered.com/?l=spanish.

[5] crontab(1) - Linux man page. https://linux.die.net/man/1/crontab.

[6] datetime – Basic date and time types – Python 3.8.2 documentation. https://docs.python.org/3/library/datetime.html.

[7] Download Android Studio and SDK tools | Android Studio. https://developer.android.com/studio.

[8] Download PuTTY - a free SSH and telnet client for Windows. https://www.putty.org/.

[9] email – An email and MIME handling package – Python 3.8.2 documentation. https://docs.python.org/3/library/email.html#module-email.

[10] email.message: Representing an email message – Python 3.8.2 documentation. https://docs.python.org/3/library/email.message.html.

[11] Enviar correo electrónico desde impresoras, escáneres o aplicaciones - Ayuda de Administrador de G Suite. https://support.google.com/a/answer/176600?hl=es.

[12] Firebase. https://firebase.google.com/?hl=es.

[13] Firebase Admin Python SDK. https://firebase.google.com/docs/reference/admin/python?hl=es.

[14] Google Play. https://play.google.com/store?hl=es_419.

[15] Java | Oracle. https://www.java.com/es/.

[16] Kotlin, declarado como lenguaje oficial por Google para desarrollos Android. https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/.

[17] Kotlin Programming Language. https://kotlinlang.org/.

[18] MySQL. https://www.mysql.com/.

[19] MySQL :: MySQL Connector/Python Developer Guide. https://dev.mysql.com/doc/connector-python/en/.

[20] MySQL :: MySQL Workbench. https://www.mysql.com/products/workbench/.

[21] Navegador web Google Chrome. https://www.google.com/intl/es_es/chrome/.

[22] Online Mockup, Wireframe & UI Prototyping Tool · Moqups. https://moqups.com/.

[23] PcComponentes.com | Tienda de Informática y Tecnología online. https://www.pccomponentes.com/.

[24] Plataforma de desarrollo colaborativo. https://github.com/.

[25] Price Tracker for Amazon - Aplicaciones en Google Play. https://play.google.com/store/apps/details?id=com.bomba.jcrocker.myapplication.

[26] Programa de afiliados PcComponentes. https://www.pccomponentes.com/soporte/cual-es-la-comision-en-el-programa-de-afiliados-de-pccomponentes.html/.

[27] Project Jupyter | Home. https://jupyter.org/.

[28] Seguidor de precio Amazon, gráficas de historial de precio de Amazon, seguimientos de precio y avisos de bajada de precios. | camelcamelcamel.com. https://es.camelcamelcamel.com/.

[29] Sistema Operativo Android. https://www.android.com/intl/es_es/.

[30] smtplib – SMTP protocol client – Python 3.8.2 documentation. https://docs.python.org/3/library/smtplib.html.

[31] Software de control de versiones. https://git.kernel.org/pub/scm/git/git.git/.

[32] Software de diagramación y creación de diagramas de flujo, Microsoft Visio. https://products.office.com/es-es/visio/flowchart-software.

[33] Stack Overflow - Where Developers Learn, Share, & Build Careers. https://stackoverflow.com/.

[34] Tema Oscuro. https://developer.android.com/guide/topics/ui/look-and-feel/darktheme.

[35] time – Time access and conversions – Python 3.8.2 documentation. https://docs.python.org/3/library/time.html.

[36] urllib – URL handling modules – Python 3.8.2rc2 documentation. https://docs.python.org/3/library/urllib.html.

[37] Welcome to Python.org. https://www.python.org/.

[38] WinSCP :: Official Site :: Download. https://winscp.net/eng/download.php.

[39] XAMPP Installers and Downloads for Apache Friends. https://www.apachefriends.org/es/index.html.

PASCAL
ENERO 2018
Ult. actualización 26 de junio de 2020
LaTeX lic. LPPL & powered by TEFLON CC-ZERO